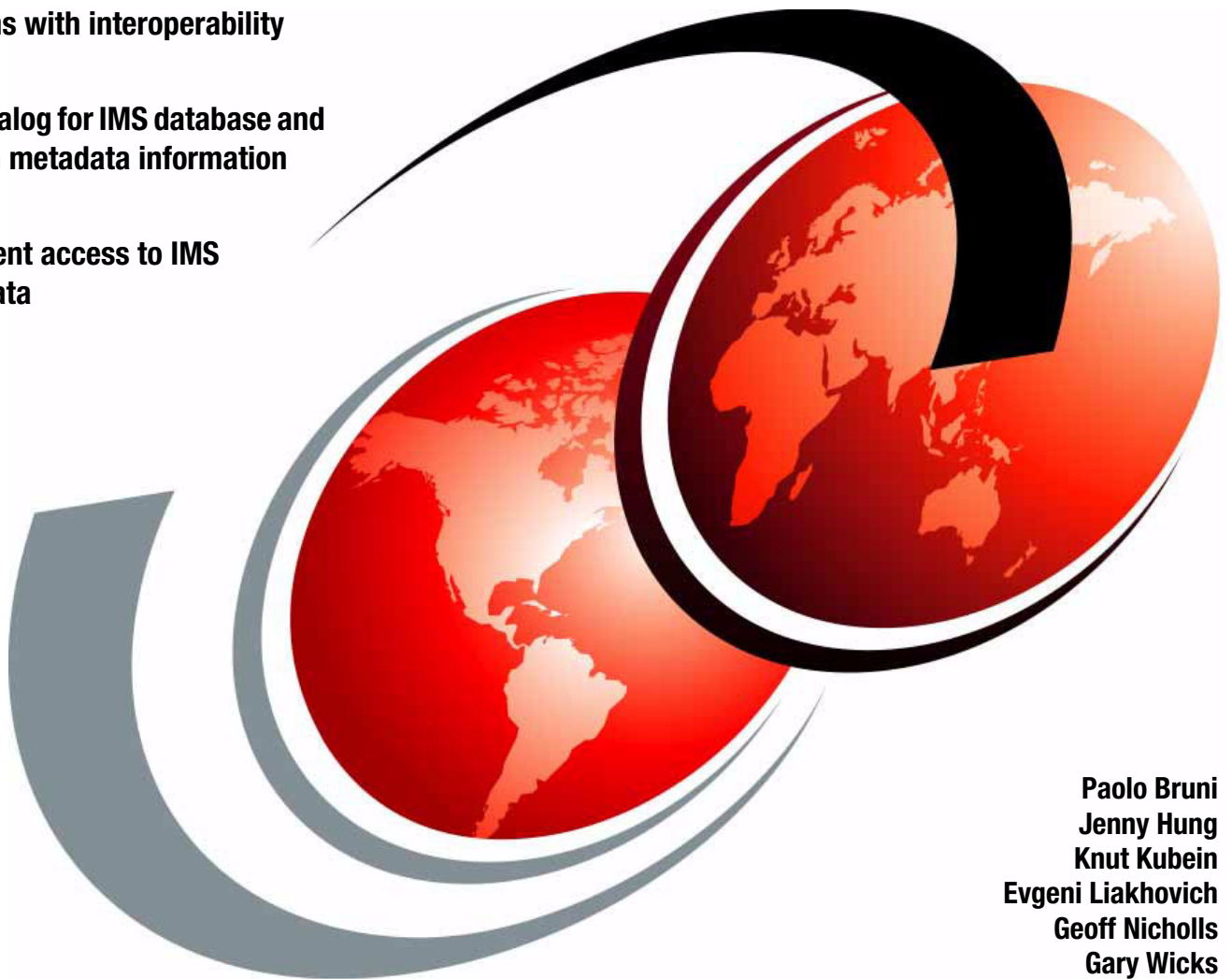


IMS Integration and Connectivity Across the Enterprise

Modernize IMS batch and online applications with interoperability

Use the catalog for IMS database and application metadata information

Expand client access to IMS and DB2 data



Paolo Bruni
Jenny Hung
Knut Kubein
Evgeni Liakhovich
Geoff Nicholls
Gary Wicks

Redbooks



International Technical Support Organization

IMS Integration and Connectivity Across the Enterprise

December 2013

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

Note: This book is based on a pre-GA version of a product and may not apply when the product becomes generally available. We recommend that you consult the product documentation or follow-on versions of this IBM Redbooks publication for more current information.

First Edition (December 2013)

This edition applies to IMS Version 12 (program number 5635-A03) and IMS Version 13 (program number 5635-A04).

© Copyright International Business Machines Corporation 2013. 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

Figures	xi
Tables	xvii
Examples	xix
Notices	xxiii
Trademarks	xxiv
Preface	xxv
Authors	xxv
Now you can become a published author, too!	xxvii
Comments welcome	xxvii
Stay connected to IBM Redbooks	xxvii
Part 1. IMS architecture and application technology fundamentals	1
Chapter 1. IMS architectural overview	3
1.1 Choosing the best IMS connectivity solution set	4
1.1.1 Considerations that are related to your choice of service	6
Chapter 2. IMS Connect	7
2.1 IMS Connect overview	8
2.2 Commit processing message flows	10
2.2.1 Commit-then-send (commit mode 0) flow	10
2.2.2 Send-then-commit message (commit mode 1) flows	11
2.2.3 Send-then-commit (sync level=none)	11
2.2.4 Send-then-commit (sync level=confirm)	12
2.2.5 Send-then-commit (sync level=sync point)	12
2.2.6 Persistent sockets	13
2.3 Installing IMS Connect	14
2.4 IMS Connect implementation and configuration process	14
2.4.1 Creating an IMS Connect start procedure	15
2.4.2 Authorizing IMS Connect and BPE to the APF	15
2.4.3 Updating the program properties table	16
2.4.4 Enabling Internet Protocol Version 6 (IPV6) for IMS Connect	16
2.4.5 Defining the IMS Connect Base Primitive Environment configuration	17
2.4.6 Base IMS Connect configuration statement parameters	18
2.4.7 Defining a single IMS Connect configuration member	28
2.4.8 Defining multiple IMS Connect configuration members	28
2.4.9 IMS Connect configuration statement for IMS DB support	30
2.4.10 Defining IMS Connect security	30
2.4.11 Installing the default user exits into an IMS Connect resource library	31
2.5 IMS Connect configuration enhancements	31
2.5.1 IMS Version 10: ACEE aging value support	32
2.5.2 IMS Version 10: Send-then-commit (CM1) ACK timeout control	32
2.5.3 IMS Version 10: Message flood control	33
2.5.4 IMS Version 10: Resume Tpipe port affinity	34
2.5.5 IMS Version 11: CM0 ACK timeout support	34
2.5.6 IMS Version 11: Super member support at a data store level	34

2.5.7	IMS Version 13: Configuring the TCP/IP maximum backlog queue size	35
2.6	IMS Connect user exits	36
2.7	IMS Connect user exit enhancements	37
2.7.1	IMS Version 11: User message exit cleanup	37
2.7.2	IMS Version 11: Modifying formats of incoming and outgoing messages.	37
2.7.3	IMS Version 11: HWSEXPRL parameter list expansion	38
2.7.4	IMS Version 12: Load modules packaging for exits	38
2.8	IMS Connect high availability facilities	38
2.8.1	IMS Connect workload balancing and failover	39
2.9	IMS Connect high availability enhancements	40
2.9.1	IMS Version 11: TCP/IP auto reconnect.	40
2.9.2	IMS Version 13: Reporting IMS Connect health to the Workload Manager	40
2.10	IMS Connect operations and command support.	42
2.10.1	IMS Connect support for IMSplex and the IMS Control Center	42
2.10.2	IMS Connect REPLY commands	43
2.10.3	IMS Connect MODIFY commands	54
2.10.4	IMS command support for IMS Connect and OTMA.	55
2.11	IMS Connect operations and command enhancements	56
2.11.1	IMS Version 11: Enhanced IMS Connect commands.	56
2.11.2	IMS Version 11: Enforcement of the single port requirement for SSL sockets.	56
2.11.3	IMS Version 12: IMS Connect type-2 SPOC commands	56
2.11.4	IMS Version 12: Partial read status display and control	60
2.11.5	IMS Version 13: IMS Connect command enhancements	61
2.11.6	IMS Version 13: Increasing the maximum number of XML converters.	61
2.11.7	IMS Version 13: Auto-restarting the Language Environment	62
2.12	Accessibility through IMS Connect	62
2.13	IMS Connect accessibility enhancements.	62
2.13.1	IMS Version 11: Integrated IMS Connect enhancements for IMS DB	62
2.13.2	IMS to IMS TCP/IP connectivity	63
2.13.3	IMS Version 12: Multiple Systems Coupling using TCP/IP.	65
2.13.4	IMS Version 13: ISC using TCP/IP protocols	67
2.14	IMS Connect client programming interfaces	72
2.15	IMS Connect Client programming interface enhancements	74
2.15.1	IMS Version 10: Alternative client ID support	74
2.15.2	IMS Version 10: Send-only with acknowledgement protocol	74
2.15.3	Cancel Client ID support.	74
2.16	IMS Connect security	75
2.16.1	IMS Connect secure access to IMS	75
2.16.2	Connecting IMS Connect to OTMA.	75
2.16.3	User verification	75
2.16.4	IMS Connect SSL connections	76
2.17	IMS Connect security enhancements	76
2.17.1	IMS Version 10: Client password change request	77
2.17.2	IMS Version 12: RACF return codes passed back to client	77
2.17.3	IMS Version 12 / 13: RACF ENF support for cached RACF user IDs.	77
2.18	IMS Connect scalability and performance.	78
2.18.1	IMS Connect performance parameters.	78
2.18.2	TCP/IP performance parameters	78
2.19	IMS Connect scalability and performance enhancements	79
2.19.1	IMS Version 10: The NODELAY parameter	79
2.19.2	IMS Version 11: IMS Connect performance enhancement.	80
2.19.3	IMS Version 12: RACF user ID caching	80
2.19.4	IMS Version 13: Usage of the internal CPOOL storage macro.	81

2.20	IMS Connect diagnostic tests	81
2.21	IMS Connect diagnostic enhancements	81
2.21.1	IMS Version 10 and 11: HWSTECLO event record enhancements	81
2.21.2	IMS Version 11: Usage of the BPE External Trace facility	81
2.21.3	IMS Version 11: Additional information in message HWSP1410W	82
2.21.4	IMS Version 11: Warning messages and early detection of maximum sockets	82
2.21.5	IMS Version 12: IMS Connect Recorder Trace	82
2.21.6	IMS Version 13: Expanded Recorder trace records	82
Chapter 3.	IMS Connect Extensions for z/OS	85
3.1	Overview	86
3.1.1	Event collection	86
3.1.2	Routing	94
3.1.3	Operational controls	96
3.2	Scenarios	97
3.2.1	Incorrect message length	97
3.2.2	Identifying IMS problems with TCP/IP information	99
3.2.3	Client fails to ACK	101
3.2.4	OTMA timeouts	103
3.2.5	Duplicate clients	105
3.2.6	Open database issues	107
3.2.7	Synchronous callout issues	112
3.2.8	OTMA flood condition	114
3.3	Scheduled and unscheduled outages	116
3.3.1	Minimizing disruptions from outages	116
3.3.2	Session rebalancing	118
3.3.3	Centrally managing client options	119
3.4	Application development considerations	120
Chapter 4.	Open Transaction Manager Access	127
4.1	OTMA clients	128
4.1.1	IMS Connect is an OTMA Client for TCP/IP	128
4.1.2	IBM WebSphere MQ on z/OS as an IMS OTMA Client	128
4.1.3	DB2 stored procedures include an OTMA Client: DSNAIMS	129
4.1.4	OTMA callable interface	130
4.2	OTMA activation, operations, and command support	130
4.3	OTMA activation, operations, and command support enhancements	131
4.3.1	IMS Version 10: Message flood condition notification	132
4.3.2	IMS Version 10: OTMA processing during IMS restart	132
4.3.3	IMS Version 10: /DIS TMEMBER Tpipe command enhanced	132
4.3.4	IMS Version 10: CM1 timeout controls	133
4.3.5	IMS Version 11: Timeout capability for commit-then-send messages	133
4.3.6	IMS Version 11: Enhanced resource monitoring	134
4.3.7	IMS Version 11: OTMA type-2 commands	135
4.3.8	IMS Version 13: /DISPLAY TMEMBER Tpipe command enhanced	135
4.3.9	IMS Version 13: OTMA global flood control enhancement	135
4.3.10	OTMA global flood control demonstration	139
4.3.11	IMS Version 13: OTMA MAXTP enhancements	143
4.3.12	IMS Version 13: MAXTP enhancement for the SQ environment	145
4.3.13	IMS Version 13: Enhancements to MAXTP support	146
4.3.14	IMS Version 13: OTMA messages sent to both the MTO and WTO	147
4.4	OTMA programming interfaces	147
4.5	OTMA programming enhancements	148

4.5.1	IMS Version 11: SQ ALTPCB back-end support	148
4.5.2	IMS Version 11: Input message transaction expiration timeout	148
4.5.3	IMS Version 12: Message DFS2082I for CM0	149
4.5.4	IMS Version 13: Synchronous Callout SendOnly Ack.	149
4.5.5	IMS Version 13: OTMA CI enhanced async support	150
4.6	OTMA security	151
4.6.1	Client bids to connect	151
4.6.2	Processing an input message from the client	151
4.6.3	Resuming transaction pipe security	152
4.6.4	OTMA callable interface security	153
4.6.5	IMS OTMA callout security	153
4.7	OTMA security enhancements	154
4.7.1	IMS Version 10: RESUME Tpipe security.	154
4.7.2	IMS Version 10: OTMA instance-specific security levels	155
4.7.3	IMS Version 12: OTMA ACEE reduction for multiple OTMA clients	155
4.8	OTMA exit and descriptor usage.	155
4.8.1	OTMA Destination Resolution user exit (OTMAYPRX).	156
4.8.2	OTMA Input/Output Edit user exit (OTMAIOED).	156
4.8.3	OTMA User Data Formatting exit routine (DFSYDRU0).	156
4.8.4	OTMA Resume Tpipe Security user exit (OTMARTUX).	156
4.8.5	OTMA descriptors	156
4.9	OTMA exit and descriptor enhancements.	157
4.9.1	IMS Version 10: OTMA Destination Routing Descriptors	157
4.9.2	IMS Version 13: DFSYPRX0 and DFSYDRU0 exits override the OTMA destination descriptor	159
4.9.3	IMS Version 13: WebSphere MQ descriptors	159
4.10	OTMA scalability and performance	161
4.10.1	Other general OTMA performance items	162
4.11	OTMA scalability and performance enhancements	162
4.11.1	IMS Version 10: Automatic removal of Tpipes	162
4.11.2	IMS Version 10: Dependent region release after ACK or NAK timeout	163
4.11.3	IMS Version 12: OTMA SQ enhancement	163
4.11.4	IMS Version 12: Reduced path length for OTMA transaction processing.	164
4.11.5	IMS Version 13: OTMA transaction expiration at GU time	164
4.11.6	IMS Version 13: OTMA ALT-PCB output for shared queues	164
4.11.7	Version 13: OTMA early termination notification.	165
4.11.8	IMS Version 13: MIPS reduction enhancement for YTIB hashing	166
4.11.9	OTMA client type notification	166
4.12	Diagnostic tests that are related to OTMA	166
4.13	OTMA diagnostic enhancements	167
4.13.1	IMS Version 11: SQ BE transaction abend error message support	167
Chapter 5.	SOAP application technology	169
5.1	IMS Enterprise Suite SOAP Gateway.	170
5.2	IMS SOAP Gateway overview	170
5.3	IMS SOAP Gateway and your IMS applications	174
5.3.1	XML-to-bytes and bytes-to-XML	175
5.4	What is new in IMS SOAP Gateway V2.2.	175
5.4.1	Advanced installation support.	176
5.4.2	The new installation architecture in IMS Enterprise Suite V2.2	177
5.4.3	Installing multiple copies of SOAP Gateway.	179
5.4.4	Monitoring, tracking, and logging	183
5.4.5	SOAP Gateway message processing events	188

5.4.6	COBOL top down and multiple hosts	189
5.4.7	SOAP Gateway support for multiple hosts for callout	192
5.4.8	WS-Security for synchronous callout and migration	194
5.4.9	IMS Enterprise Suite V3.1 features	197
5.5	Asynchronous callout with SOAP Gateway	199
5.5.1	Starting the web service operation	199
5.5.2	Returning the callout response message to IMS	200
5.5.3	Web services callout scenarios for Asynchronous Callout	200
5.6	Synchronous callout with IMS SOAP Gateway	207
5.6.1	Synchronous callout support with IMS SOAP Gateway	207
5.6.2	Synchronous callout user scenarios	209
5.7	Multisegment support	212
5.8	Security enhancements	213
5.9	IMS SOAP Gateway features and compatibilities	214
Chapter 6.	Java Platform, Enterprise Edition application technology	215
6.1	Java Platform, Enterprise Edition	216
6.1.1	Developing Java applications	216
6.1.2	IBM Rational Developer for System z	220
6.1.3	The IBM IMS TM Resource Adapter	224
6.2	Key features of the IMS TM Resource Adapter	225
6.2.1	Features that are introduced in IMS TM Resource Adapter Version 10.2	226
6.2.2	Features that are introduced in IMS TM Resource Adapter Version 11	227
6.2.3	New features in IMS TM Resource Adapter Version 12	228
6.3	Installing the IMS TM Resource Adapter	232
6.4	Callin request support	235
6.5	Callout request support	238
6.5.1	Destinations for callouts through IMS TM Resource Adapter	239
6.5.2	Asynchronous callout requests	240
6.5.3	Synchronous callout requests	244
6.5.4	Issuing synchronous callout requests from a Java dependent region	252
6.5.5	IMS Enterprise Suite Connect APIs	254
6.5.6	IBM IMS DB Resource Adapter (DB Universal Driver)	255
6.5.7	IMS Universal drivers: Configuring connections to IMS	256

Part 2. Extended architecture and application technology of IBM IMS 261

Chapter 7.	COBOL dynamic SQL access to IBM IMS databases	263
7.1	Overview	264
7.2	SQL statements that are supported by IMS	264
7.3	Sample IMS COBOL SQL program	265
Chapter 8.	The IMS catalog	273
8.1	Overview and objectives of the catalog	274
8.2	Physical structure of the catalog database	276
8.2.1	Segments of the catalog database	276
8.3	IMS catalog database installation and management	281
8.3.1	Installation	282
8.3.2	IMS catalog initial data population	284
8.3.3	ACB generation and changes	286
8.3.4	IMS Catalog Copy utility	286
8.3.5	Keeping multiple versions of metadata in the catalog	287
8.3.6	IMS Catalog Record Purge utility	288
8.3.7	Automatically creating the IMS catalog database data sets	289

8.3.8	Using the IMS catalog without DBRC	289
8.3.9	Aliases and sharing.	290
8.3.10	Definitions that are needed for the IMS catalog	293
8.4	Application usage of the catalog	293
8.4.1	DBD and PSB source changes.	294
8.4.2	Get Unique Record DL/I call	298
8.4.3	IMS catalog access.	302
8.4.4	SSA enhancements	303
8.5	The role of the IMS Enterprise Suite Explorer for Development	304
8.5.1	Extending IMS database definitions with the IMS Explorer.	307
8.6	Using IMS Explorer to capture IMS metadata.	307
8.7	Enhancements to the IMS Universal drivers	330
8.7.1	Access to the IMS databases from Java.	330
8.7.2	Using the metadata information in the DL/I access.	334
Chapter 9.	IMS Explorer and Open Database connectivity	347
9.1	IMS Explorer for Developers	348
9.2	How IMS Enterprise Suite extends access to IMS	351
9.2.1	IMS Explorer demonstration	356
9.2.2	Creating or modifying full-function or Fast Path database PCBs	359
9.2.3	Connecting to an IMS database for SQL access	360
9.2.4	Establishing the host connection with IMS Explorer for Developer.	361
9.2.5	Creating and running SQL queries against an IMS database.	363
Chapter 10.	IMS Data Provider for Microsoft .NET	367
10.1	Overview of the IBM IMS Data Provider for Microsoft .NET	368
10.1.1	ADO.NET application development	368
10.1.2	IMS Data Provider prerequisites and system requirements	369
10.1.3	IMS Data Provider architecture.	369
10.2	Programming applications with IBM IMS Data Provider for Microsoft .NET	371
10.2.1	Generic coding with the ADO.NET common base classes.	371
10.2.2	Connecting to IMS from an application.	372
10.2.3	Connection pooling with the IMS Data Provider	373
10.2.4	SQL data type representation in ADO.NET database applications.	373
10.2.5	Running SQL statements	374
10.2.6	Reading result sets	376
10.2.7	Connected versus Disconnected Modes.	376
10.2.8	Choosing between DataReader and DataSet.	376
10.2.9	Using IMSDataAdapter and DataSet for Disconnected data processing	377
10.2.10	Processing metadata with IMS Data Provider.	377
10.2.11	Tracing IMS Data Provider	380
Chapter 11.	IMS mobile enablement solutions.	383
11.1	IMS mobile enablement	384
11.2	IBM Worklight	385
11.2.1	Accessing IMS transactions through Worklight HTTP Adapter.	385
11.2.2	Accessing IMS data through the Worklight SQL adapter	388
11.2.3	Using a custom Java application in Worklight to connect to IMS	389
11.3	IBM WebSphere DataPower.	390
11.3.1	RESTful service facade	391
11.3.2	DMZ Proxy to secure a mobile network	392
11.3.3	Seamless enterprise integration for IBM Worklight.	392
Chapter 12.	IBM WebSphere DataPower and IMS integration.	395

12.1	WebSphere DataPower SOA Appliances introduction	396
12.2	WebSphere DataPower capabilities	398
12.2.1	Any-to-any transformation engine	398
12.2.2	Control	398
12.2.3	Domains	398
12.2.4	Firewall	399
12.2.5	Firmware updates	399
12.2.6	Interfaces	399
12.2.7	Logging	399
12.2.8	Optimization	399
12.2.9	Monitoring	399
12.2.10	Multi-Protocol Gateway	399
12.2.11	WS-Proxy	400
12.2.12	XML management interface	400
12.2.13	XML manager	400
12.2.14	XML processing	401
12.2.15	XSL co-processor	401
12.3	WebSphere DataPower and IMS integration solutions	401
12.3.1	Requirements for inbound access to IMS transactions	401
12.3.2	Requirements for IMS Synchronous Callout support	402
12.3.3	Requirements for access to the IMS database	402
12.3.4	Inbound access to IMS transactions from an external client	402
12.3.5	Outbound support from IMS synchronous callout to an external client	412
12.3.6	Access to IMS databases	421
Part 3.	Appendixes	427
	Appendix A. Sample code	429
A.1	ICAL Synchronous Program Switch COBOL program	430
A.2	Asynchronous callout to a StateLess Session bean	435
A.3	Asynchronous callout to a Message Driven bean	437
A.4	Synchronous callout to a StateLess Session bean	440
A.5	Synchronous callout to a Message Driven bean	442
A.6	Feed from an IMS application in MashupHub	443
A.7	WSDL files for a DLIModel generated web service	445
A.8	XSD files for a DLIModel generated web service	446
A.9	Enhanced Provider MPP template sample	451
	Appendix B. Additional material	461
	Locating the web material	461
	Using the web material	461
	System requirements for downloading the web material	461
	Downloading and extracting the web material	462
	Related publications	463
	IBM Redbooks	463
	Other publications	463
	Online resources	464
	Help from IBM	464
	Index	465

Figures

2-1	IMS Connect overview	8
2-2	Commit-then-send message flows	10
2-3	Send-then-commit message flow with sync level=none, deallocate confirm	11
2-4	Send-then-commit message flow with sync level=confirm	12
2-5	Send-then-commit message flow with sync level=sync point	13
2-6	Simple IMS Connect configuration	28
2-7	Configuration for multiple IMS Connects servicing multiple IMS images	29
2-8	Simple IMS Connect system configuration for IMS DB support	30
2-9	Sample super member	35
2-10	IMS Connect workload balancing and failover	39
2-11	IMS Connect and the WLM Health Report	41
2-12	Overview of IMS Connect support for IMS Control Center	42
2-13	Output of the type-2 QUERY IMSCON TYPE(PORT) command from the SPOC	58
2-14	Asynchronous IMS to IMS communication through TCP/IP	63
2-15	Asynchronous IMS to IMS communication through TCP/IP details	64
2-16	MSC using TCP/IP for IMS Connectivity	65
2-17	Overview of the ISC communication solution over Internet Protocol networks	68
2-18	ISC communication over TCP/IP functionality	68
2-19	Initiating transactions using an ISC TCP/IP connection	69
2-20	Usage of CSL in support of the IMS and IMS Connect ISC routing components	70
2-21	IMS Connect configuration sample that is associated with ISC support	71
2-22	ISC TCP/IP component interface	71
2-23	Message flow between a TCP/IP client and IMS (OTMA)	73
3-1	IMS Connect Extensions journals combine with the IMS log to offer information about TCP/IP transactions	87
3-2	IMS Connect Extensions also records Open Database requests	88
3-3	IMS Problem Investigator tracks all the records for a transaction from across IMS Connect, IMS, and DB2	89
3-4	Without IMS Connect event recording there is a performance gap that can stifle cooperation between groups	90
3-5	IMS Connect Extensions is required to gain an end-to-end report of TCP/IP transaction performance	91
3-6	Transit Analysis report fields for sync level none transactions	92
3-7	Transit Analysis report fields for sync level confirm transactions	93
3-8	Routing configuration and the corresponding result	95
3-9	Shaping routing for different network demand	96
3-10	Centrally manage IMS Connect instances in ISPF or using the GUI	97
3-11	Obtaining the event key for a hanging session in the GUI	98
3-12	The difference between the request length and actual length indicates an incorrect message length	98
3-13	IMS Connect transaction index that is merged in IMS Problem Investigator	99
3-14	Search for long transactions from an IP address	100
3-15	Initiate tracking from the transaction index that is identified by the filter	100
3-16	Both the IMS and IMS Connect log records for the transaction (tracked from index)	101
3-17	Session error in IMS Problem Investigator logs	102
3-18	Protocol violation error appearing in the log record	103
3-19	Reporting timeouts in IMS Performance Analyzer	104
3-20	Identifying the timeout value from the IMS Connect Extensions journal	104

3-21	Overriding the ACK/NAK timeout for a transaction code	105
3-22	Duplicate client error in IMS Connect Extensions journal	106
3-23	Overriding the Client ID cancellation option for a transaction code	106
3-24	Filtering by A047 records	107
3-25	Displaying the message area	108
3-26	Tracking the complete flow	109
3-27	DRDA and DL/I flow	110
3-28	Displaying detailed information by pressing F11	111
3-29	Displaying the I/O area	112
3-30	Synchronous callout reporting in IMS Performance Analyzer	112
3-31	Synchronous callout event flow in IMS Problem Investigator	113
3-32	OTMA flood condition in the IMS Connect Extensions GUI	114
3-33	Issuing a command to increase the flood threshold from the IMS Connect Extensions GUI	115
3-34	Outages at different subsystems require different responses and coordination	116
3-35	Route Drain suspends the data store and prevent new activity but allows existing sessions to complete	117
3-36	Stop the data store when there are no sessions and start the data store when maintenance completes	117
3-37	Initiate a data store drain in batch	118
3-38	Query the status of a data store in batch	118
3-39	Configuring session rebalancing	119
3-40	Centrally-configurable client options	120
3-41	Common features for client/server development	121
3-42	An IDE extended with IMS Connect Extensions	122
3-43	Diagnosing an SQL call failure	123
3-44	Open database (ODBM) sessions	124
3-45	Initiating a trace	125
3-46	Browsing the journal records for an Open Database request	126
4-1	DSNAIMS stored procedure for OTMA C/I access	129
4-2	OTMA and its interfaces	130
4-3	Global flood control enhancement	136
4-4	Commands and status that is related to GFC	137
4-5	OTMA Client Descriptor DFSOTMA that is used to set the GFC value	138
4-6	Setting up the MAXTP parameter in OTMA client descriptor in DFSYSSTx	143
4-7	Monitoring actions and messages for a single OTMA member	144
4-8	Monitoring for all members that have MAXTP	145
4-9	IMS 12 Synchronous Callout SendOnly Ack SPE	150
4-10	Details of the D descriptor type	158
4-11	OTMA descriptor and asynchronous call messages to WebSphere MQ	160
4-12	IMS Version 13 WebSphere MQ OTMA Descriptor example	161
4-13	IMS type-2 OTMADESC commands	161
4-14	Override an SQ back-end affinity for ALTPCB messages	165
5-1	SOAP Gateway	170
5-2	New Management Utility	173
5-3	Flow of control with an IMS application as a web services provider	174
5-4	Before Enterprise Suite V2.2	177
5-5	The new architecture	178
5-6	SMP/E installation process	179
5-7	Multiple copies of SOAP Gateway	180
5-8	deploy>./iogmgmt -view -sgp output	182
5-9	IMS Transaction tracking – Provider scenario	185
5-10	SOAP Gateway log of an input message	186

5-11	ICONTR entry of a BPE trace	187
5-12	Transaction log	189
5-13	Design flow	190
5-14	Operational considerations	191
5-15	Multiple hosts	193
5-16	Run time of a synchronous callout processing	195
5-17	Error scenario #1	197
5-18	Error scenario #2	197
5-19	Asynchronous Callout flow	199
5-20	SOAP callout to a one-way web service invocation with no response	201
5-21	Callout message after OMTA descriptor processing.	201
5-22	Callout message with XMLAdapterOutput tag	202
5-23	Callout message with SOAP tag	202
5-24	SOAP callout to a one-way web service invocation with a response	203
5-25	Message flow of SOAP callout to a request-response web service invocation.	205
5-26	IMS Synchronous Callout support	207
5-27	Overview of the IMS Synchronous Callout flow	208
5-28	Callout flow with multiple IMS applications	209
5-29	Callout flow with multiple IMS Connects	210
5-30	Callout flow with IMS shared queues	211
5-31	Callout flow with multiple servers	211
5-32	SSL/HTTP message flow	213
6-1	HelloWorld	218
6-2	Java Editor example	219
6-3	Cheat sheets.	220
6-4	Rational Developer for System z	221
6-5	Enhanced Provider MPP template overview.	223
6-6	WebSphere Transformation Extender support	227
6-7	Run Administrative Console	233
6-8	Resource adapter configuration in WebSphere Application Server	234
6-9	Common Client Interface contract	235
6-10	Custom properties of IMS Connection Factory	236
6-11	ConnectionSpec and InteractionSpec.	237
6-12	Callout data flow	239
6-13	Message flow of asynchronous callout to SLSB using IMS TMRA.	242
6-14	Message flow of asynchronous callout to MDB using IMS TMRA	243
6-15	Synchronous callout interfaces	245
6-16	Message flow of a synchronous callout to SLSB using IMS TMRA	247
6-17	Message flow of a synchronous callout to MDB using IMS TMRA	248
6-18	Supported software configurations by version of the IMS TMRA	251
6-19	z/OS UNIX mount point sample	257
6-20	Type 2 and Type 4 Connectivity	260
8-1	A PSB with multiple PCBs	274
8-2	Database structure for the catalog database DFSCD000.	280
8-3	IMS Catalog Populate utility (DFS3PU00)	285
8-4	ACB Generation and IMS Catalog Populate utility	286
8-5	Multiple IMS, cloned ACBLIB, and shared IMS catalog	291
8-6	Multiple IMS, cloned ACBLIB, and one IMS catalog for each IMS	292
8-7	Multiple IMS, shared ACBLIBs, and shared IMS catalog	292
8-8	Several mappings for the same segment	295
8-9	IMS catalog access.	303
8-10	Using get by offset	304
8-11	Field sensitivity	304

8-12	IMS Explorer and catalog	305
8-13	Export to the z/OS host with FTP	306
8-14	IMS Explorer perspective in Rational Developer for zEnterprise	308
8-15	Create a New Project wizard	309
8-16	IMS Explorer Project	309
8-17	Name the IMS Explorer Project	310
8-18	The LGI Application IMS Explorer Project	311
8-19	Options for an IMS Explorer Project	312
8-20	Select an import source	313
8-21	Provide a name for the IMS Resources to be imported	314
8-22	Select the Import Source of the source to be imported.	315
8-23	Select resources from the local file system.	316
8-24	Select the file that contains the resources to be imported	317
8-25	Keep selecting more files until you are done	318
8-26	Selecting the PSB source to be imported	319
8-27	Lists of the DBD and PSB resources to be imported	320
8-28	LGI Databases project showing the DBD and PSB resources that are imported	321
8-29	Expanding the DBD Resources view	321
8-30	The IMS Explorer view of an IMS database	322
8-31	Importing database metadata from COBOL or PL/I	323
8-32	Select the file that contains the copybook or include member	323
8-33	Identify the data structure to be imported for a database segment.	324
8-34	Importing the metadata for a database segment	325
8-35	IMS Explorer database view with imported metadata for one segment	326
8-36	Explorer database view with imported metadata for all segments	327
8-37	The catalog-enabled DBD source.	328
8-38	Sample of a DBD with catalog-enabled source	329
8-39	Connection of a Java program through the drivers.	331
8-40	Insurance segment mapped multiple ways based on the Policy Type control field	340
8-41	Segment view on disk.	340
9-1	Simplification strategy	348
9-2	SDFSISRC(DFS AUTDB)	350
9-3	AUTODB DBD Source	351
9-4	Showing the logical relationship	352
9-5	Type of relationships in the Properties view	353
9-6	Typical PCB definition within a PSB	353
9-7	Display and build the PSB and all the PCBs.	354
9-8	Table update of the Phonebook DB	355
9-9	JDBC Trace	356
9-10	Select Import Source	357
9-11	Select a connection profile	357
9-12	Available PSBs to import	358
9-13	Modify or add data	359
9-14	Unable to Perform Function	361
9-15	Connection Parameters	362
9-16	UNIX file view by using Rational Developer for System z.	363
9-17	SQL Query Builder	364
9-18	Query result	364
9-19	SQL to DL/I translator	365
9-20	Delete row.	366
10-1	Overview of IMS .NET Data Provider architecture	370
11-1	Overview of the connectivity options for IMS mobile enablement.	384
11-2	Mobile connectivity through Worklight Server and IMS SOAP Gateway	386

11-3	Accessing IMS from mobile devices through the Worklight SQL adapter.	388
11-4	IMS mobile enablement with WebSphere DataPower	391
11-5	Mobile scenario with WebSphere DataPower starting an IMS application issuing an outbound callout for a web service.	392
12-1	WebSphere DataPower common use cases	397
12-2	Inbound data flow to IMS as a service provider	403
12-3	WebSphere DataPower Control Panel	405
12-4	WebSphere DataPower Multi-Protocol Gateway configuration.	406
12-5	Advanced configuration in Multi-Protocol Gateway	407
12-6	Adding a front-side handler to the Multi-Protocol Gateway.	408
12-7	Configuring a Multi-Protocol Gateway Policy with rules and actions	409
12-8	Creating an IMS Connect configuration	410
12-9	Defining information for the IMS Connect object	411
12-10	Outbound data flow from IMS as service consumer.	413
12-11	Configuring IMS Callout Front Side Handler.	417
12-12	Advanced IMS Callout configuration.	419
12-13	Configuring SQL Data Source	423
12-14	Additional parameters for Data Source Configuration	424
A-1	ActivationSpec.	438
A-2	ActivationSpec properties.	439
A-3	ActivationSpec name in Deployment Descriptor of MDB	440

Tables

1-1 Technologies and products that are associated with IMS connectivity solutions	4
2-1 IMS Connect user exits that are available at the IMS Version 13 level.	36
2-2 Equivalence between REPLY and MODIFY commands.	54
2-3 IMS type-2 Display commands	57
2-4 IMS Connect type-2 start commands	58
2-5 IMS Connect type-2 stop or close commands	59
2-6 IMS Connect type-2 set, reset, and refresh commands	59
2-7 Commands for User ID caching	80
2-8 IMS Connect additional recorder trace entries	82
5-1 The different paths	183
5-2 Summary of IMS SOAP Gateway features and compatibilities.	214
6-1 Properties for the IMSActivationSpec object.	229
6-2 com.ibm.connector2.ims.ico class summary.	250
6-3 IMS Universal resource adapter comparison	258
9-1 Statement properties.	360
10-1 Mappings across IMS, DbType, and Microsoft .NET Framework types	374
10-2 GetSchemaTable method results	378
10-3 GetSchemaTable method results	379

Examples

2-1	IMS Connect startup JCL	15
2-2	APF entries for IMS systems in SYS1.PARMLIB(PROGxx)	16
2-3	PPT entry that is required for IMS Connect	16
2-4	Sample BPE configuration file	17
2-5	Sample configuration file.	28
2-6	IMS Connect configuration for three IMS Connect instances	29
2-7	Sample IMS Connect configuration definitions for IMS DB support	30
2-8	Sample JCL to install the default user exits	31
2-9	VIEWHWS showing the ACEE aging value	32
2-10	Sample IMS Connect configuration member HWCF13B1	43
2-11	An example of the CLOSEHWS command.	46
2-12	Example messages that are associated with an IMS Connect initialization	47
2-13	An example of the OPENDS command	47
2-14	Output of the OPENIP command	48
2-15	Output of the OPENPORT command	48
2-16	Output of the RECORDER command.	49
2-17	An example of the SETRACF command	49
2-18	An example of the SETRRS command.	49
2-19	An example of the STOPCLNT command	50
2-20	An example of the STOPDS command	50
2-21	An example of the STOPIP command	51
2-22	An example of the STOPPORT command	51
2-23	An example of the VIEWDS command.	51
2-24	An example of the VIEWHWS command	51
2-25	An example of the VIEWIP command.	53
2-26	An example of the VIEWPORT command	53
2-27	Output of the IMS /DISPLAY OTMA command	55
2-28	Output of the /DISPLAY TMEMBER ALL command.	55
2-29	Output from the VIEWPORT command with a partial read client	60
2-30	The STOPCLNT command.	61
2-31	IMS stage 1 macros for a TCP/IP MSC link (local system).	66
2-32	IMS stage 1 macros for a TCP/IP MSC link (remote system).	66
2-33	IMS Connect configuration for a TCP/IP MSC link (local system)	66
2-34	IMS Connect configuration for a TCP/IP MSC link (remote system).	66
2-35	Output of the SETUIDC ON command	77
2-36	Sample IMS Connect configuration member with the RACF user ID caching specification	80
3-1	IMS Problem Investigator display of key fields in an index record	89
3-2	Transit Analysis report for sync level none transactions.	92
3-3	Transit Analysis report for sync level confirm transactions.	92
3-4	Transit Analysis report by transaction and one-minute time with subtotals	93
3-5	Output of data store batch query	118
4-1	IMS Connect configurations: HWSCFG01 HWSCFG02 and HWSCFG03.	139
4-2	Setting the INPT value of 3 in the OTMA descriptor DFSYDTC.	140
4-3	Modifying the DFSYDTC member with an invalid INPT value of 999999.	140
4-4	OTMA flood indication and action messages	141
4-5	/DIS OTMA associated with SERVER+FLOOD status	141
4-6	/START TMEMBER ALL INPUT 0	142

5-1	Connection bundle example	171
5-2	Correlator file example for IMS SOAP Gateway	172
5-3	AEWTSGCF sample	180
5-4	Mount point imsserver/deploy on OMVS	181
5-5	Monitoring that is sent to a specific port	184
5-6	Sample commands for tracking ID	184
5-7	Configure and enable transaction file logging	187
5-8	xsebatch that you need	191
5-9	DFSYTDx sample	193
5-10	Connection bundles	193
5-11	Update the correlator	193
5-12	Deploy the service	193
5-13	Update client side	196
5-14	Deploy SAML11 token	196
5-15	Debug mode	196
5-16	Check certificate	197
6-1	GET-HOLD-NEXT snippet	223
6-2	OTMA descriptor keyword SYNTIMER	239
6-3	Sample IMSQueueConnectionFactory	253
7-1	Sample IMS COBOL SQL program	265
8-1	Segments in the IMS catalog database (DFSCD000)	277
8-2	Copy the supplied DBD and PSB members to your own libraries	282
8-3	ACBGEN for IMS catalog	282
8-4	Define the HALDB data sets, the ILDS, and the primary and secondary indexes	282
8-5	IMS.PROCLIB(DFSDFxxx) for a single IMS system	283
8-6	IMS.PROCLIB(DFSDFxxx) for multiple IMS systems	284
8-7	Running the IMS Catalog Populate utility (DFS3PU00)	285
8-8	Setting default maximum generations and retention periods for catalog metadata	288
8-9	IMS catalog parameters in DFSDFxxx	289
8-10	Example SYSIN for DFS3UCD0	290
8-11	IMS.PROCLIB(DFSDFxxx) definition for a non-DBRC IMS catalog	290
8-12	DFSMDA definition	290
8-13	Single IMS system	293
8-14	Multiple IMS systems	293
8-15	Sample application that uses the GUR DL/I call	299
8-16	Sample data that is returned by a GUR call	301
8-17	Sample data that is returned by a browser	301
8-18	DFSDDLTO statements for GUR	302
8-19	Obtain a connection with IMSDataSource for use with JDBC	331
8-20	Obtain a connection with PSB for use with DL/I	333
8-21	Variable segment length information in DatabaseView class	335
8-22	Variable segment length information in IMS catalog: XML	335
8-23	COBOL structure example	335
8-24	Structure information in the IMS catalog	336
8-25	JDBC SQL for structure retrieve	336
8-26	Updating the PERSON segment with JDBC	337
8-27	DL/I access for structure retrieve	337
8-28	COBOL array	338
8-29	Retrieve of the segment STUDENT	338
8-30	Insertion of a new STUDENT segment	339
8-31	IMS Universal drivers pull metadata from the IMS catalog GUR call as XML	340
8-32	Cases and mapping	341
8-33	COBOL structure example	342

8-34	Redefine metadata information in the IMS catalog	342
8-35	Field with a fully defined converter class	343
8-36	User-defined type converter for the PackedDate type	343
8-37	Example of an explicit GUR call	344
8-38	Example of a getCatalogMetaDataAsXML public method definition.	344
9-1	SQL insert sample	355
10-1	Simple C# application connecting to IMS and retrieving data.	370
10-2	Generic ADO.NET approach	372
10-3	IMSPParameter and Prepare methods in C#	373
10-4	SELECT statement in C#	375
10-5	UPDATE statement in C#	375
10-6	Rolling back or committing a transaction in C#.	375
10-7	Reading a result set in C#.	376
10-8	Using the GetSchema method to display database schema information	379
10-9	Trace output	380
11-1	HTTP adapter configuration file (IMSAdapter.XML)	386
11-2	JavaScript file that is configured to connect to the IMS SOAP Gateway based Phonebook web service	387
11-3	Response from the JavaScript file	387
11-4	JavaScript file that is configured to query an IMS database.	388
11-5	Java code that scans an IMS catalog	389
11-6	Referencing the custom Java code.	390
12-1	Linking the HWSDPWR1 exit routine	415
12-2	Sample XSL	420
12-3	Sample XSL with extension element	424
A-1	Source and assembly plus bind jobs for COBOL ICAL Synchronous Program Switch program IAPMDI27.	430
A-2	Code snippet for asynchronous to SLSB in WebSphere Application Server	435
A-3	Code snippet for asynchronous callout to MDB in WebSphere Application Server . .	437
A-4	Code snippet for synchronous callout to SLDB in WebSphere Application Server . .	440
A-5	Code snippet for synchronous callout to MDB in WebSphere Application Server . .	442
A-6	Example of an IMS feed in MashupHub.	443
A-7	DealersModels.wsdl	445
A-8	AUTPSB11-AUTS2PCB.xsd	446
A-9	Enhanced Provider MPP template sample.	451

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 grant 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites 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.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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®	Rational®
CICS®	IMS™	Redbooks®
CICS Explorer®	InfoSphere®	Redbooks (logo)  ®
Cognos®	Language Environment®	System z®
DataPower®	MVS™	System z10®
DataStage®	OMEGAMON®	VTAM®
DB2®	Optim™	WebSphere®
Distributed Relational Database Architecture™	OS/390®	z/OS®
DRDA®	Parallel Sysplex®	z10™
	RACF®	zEnterprise®

The following terms are trademarks of other companies:

Worklight is trademark or registered trademark of Worklight, an IBM Company.

Intel, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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

Preface

This IBM® Redbooks® publication gives a broad understanding of IBM IMS™ integration and connectivity solutions to access applications and data stores across your enterprise architecture.

As an application developer, architect, systems integrator, or systems programmer, there is important information that is available in this book that pertains to your responsibilities to continue to include the proven performance, data integrity, and workload distribution that is available from IMS in to selected projects that are related to your entire enterprise.

This book updates and adds to the information in the following IBM Redbooks publications:

- ▶ *IMS e-business Connectors: A Guide to IMS Connectivity*, SG24-6514
- ▶ *IMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity*, SG24-6794
- ▶ *Powering SOA Solutions with IMS*, SG24-7662
- ▶ *IBM IMS Version 12 Technical Overview*, SG24-7972
- ▶ *IMS 12: The IMS Catalog*, REDP-4812
- ▶ *Rethink Your Mainframe Applications: Reasons and Approaches for Extension, Transformation, and Growth*, REDP-4938

Authors

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

Paolo Bruni is an ITSO Project Leader that is based at the Silicon Valley Lab in San Jose, CA. Since 1998, Paolo has authored IBM Redbooks publications about IMS, DB2® for z/OS®, and related tools, and has conducted workshops worldwide. During his many years with IBM in development and in the field, Paolo's work has been related mostly to database systems.

Jenny Hung is an advisory software engineer working in IMS OnDemand to modernize IMS as the integration focal point in a service-oriented architecture (SOA) environment. She is actively involved in helping IMS customers to make the SOA transition. Her expertise includes the IBM WebSphere® DataPower® IMS integration, IMS Enterprise Suite SOAP Gateway, IMS Connect API, IMS Transaction Manager (TM) Resource Adapter, and IMS Business Process Choreography. Jenny has a Master's degree in Computer Science from Stanford University. Jenny's work relates mostly to the web and transaction processing.

Knut Kubein is a Senior Advisory I/T Specialist with IBM Global Services in Germany. He has 43 years of experience with IBM large systems products working as a Technical Support Specialist, with 33 of those years devoted to IMS. He leads the IMS Support Team in EMEA. His areas of expertise include IMS data sharing, shared queues, and IBM Parallel Sysplex® architecture. He supports large banking clients and other industrial IMS users. Knut co-authored several IBM Redbooks publications on Parallel Sysplex, performance with IMS, and powering SOA solutions with IMS.

Evgeni Liakhovich is an Advisory Software Engineer at IBM. He has a Master's degree in Computer Science and has been working for IBM for 14 years. In 2006, he joined the IMS team as a support engineer for eBusiness OnDemand products. He always had great interest in IMS modernization and in 2008 he joined the IMS SOA team. He has created IMS educational material and presented at customer seminars and road shows. He is also part-time developer, leading one of the next-generation modernization projects for IMS.

Geoff Nicholls is a Consulting IT Specialist, working in the IMS Solution Test team for the IBM Silicon Valley Laboratory, and is based in Melbourne, Australia. Before his current role, Geoff was a member of the Worldwide IMS Advocate team for 12 years, providing consulting, education, and services to clients in many industries around the world. Geoff is the co-author of 13 IMS IBM Redbooks publications. Before joining IBM, Geoff worked as an Applications Programmer and Database Administrator for several insurance companies and another mainframe vendor. Geoff has a Bachelor of Science, majoring in Computer Science, from the University of Melbourne.

Gary Wicks is a Certified Board Level IT Specialist with the IMS World Wide Specialists Group in San Jose California. He has 40 years of experience with IBM large system software support, with 34 years in the IMS field. He holds a degree in Physics from the University of Toronto from 1973. His areas of expertise include supporting customers worldwide for performance and implementation designs of IMS data sharing, shared queues, and the Parallel Sysplex architecture. He has written extensively on IMS data sharing, Multiple Systems Coupling, IMS and SOA, and on-demand architectures.

Special thanks to Fundi Software representative Rafael Avidad for his exceptional support during this project.

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

Bob Haimowitz

IBM International Technical Support Organization

Dave Cameron

Kyle Charlet

Chih-Fang Li

Peter Miotke

Tom Morrison

Shyh-Mei Ho

Richard Tran

Jack Yuan

IBM Silicon Valley Laboratory

Dougie Lawson

IBM UK

Ken Blackman

Suzie Wendler

IBM Dallas ATS

Rafael Avigad

James Martin

Fundi Software

Now you can become a published author, too!

Here's an opportunity to spotlight your skills, grow your career, and become a published author—all at the same time! Join an ITSO residency project and help write a book in your area of expertise, while honing your experience using leading-edge technologies. Your efforts will help to increase product acceptance and customer satisfaction, as you expand your network of technical contacts and relationships. Residencies run from two to six weeks in length, and you can participate either in person or as a remote resident working from your home base.

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

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

- Send your comments in an email to:

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

Stay connected to IBM Redbooks

- Find us on Facebook:

<http://www.facebook.com/IBMRedbooks>

- Follow us on Twitter:

<http://twitter.com/ibmredbooks>

- Look for us on LinkedIn:

<http://www.linkedin.com/groups?home=&gid=2130806>

- Explore new Redbooks publications, residencies, and workshops with the IBM Redbooks weekly newsletter:

<https://www.redbooks.ibm.com/Redbooks.nsf/subscribe?OpenForm>

- Stay current on recent Redbooks publications with RSS Feeds:

<http://www.redbooks.ibm.com/rss.html>



Part 1

IMS architecture and application technology fundamentals

The IMS environment is one that can seamlessly integrate the computing infrastructure (hardware, software, and the related integration services) with vital business applications. It has the following components:

- ▶ Integrated

The goal is to allow business applications to be able to interoperate end-to-end across your enterprise. This means integration between IMS and its controlled data stores with other products within IBM and with other products and platforms that are available to the industry.

- ▶ Open

IMS applications have the flexibility to access mainframe-based data stores and call out to middleware platforms to meet your business needs.

- ▶ Virtualized

Well-configured IMS systems autonomically handle transaction throughput rate increases to create cost-efficiency and maximize your IT investment. This provides system controlled virtualization for the flexibility to grow and expand, and the ability to use new resources as they become available.

- ▶ Autonomic computing

For example, with the use of automation and available threshold controls, IMS can be configured to manage flood conditions, allowing you to focus on your business, not the specifics of your selected technology implementations.

IMS addresses all four of these components. IBM provides support for various connectivity and integration solutions with IMS. IMS views integration as a continuing journey and continues to support and enhance new technology for connectivity and e-business enablement into the foreseeable future.

This part includes the following chapters:

- ▶ Chapter 1, “IMS architectural overview” on page 3
- ▶ Chapter 2, “IMS Connect” on page 7
- ▶ Chapter 3, “IMS Connect Extensions for z/OS” on page 85
- ▶ Chapter 4, “Open Transaction Manager Access” on page 127
- ▶ Chapter 5, “SOAP application technology” on page 169
- ▶ Chapter 6, “Java Platform, Enterprise Edition application technology” on page 215



IMS architectural overview

As IMS celebrates its 45th year anniversary in 2013, the IMS development laboratory in San Jose, California and, by extension, IBM IMS worldwide support teams, look forward to the international client base continuing to integrate their IMS assets into their modern enterprises.

IT architects and application design and development teams have many opportunities available to them that are related to modernization and enterprise integration challenges. The key challenge is developing solutions that combine stability, high availability, capacity, and performance while simplifying application development and reducing the total cost of ownership (TCO).

There are many components that are available to IMS based environments that allow connectivity from distributed and mainframe-centric applications to IMS and its databases, and also mainframe IMS applications to call out to obtain information from various data stores. This chapter maps out the various integration solutions for IMS connectivity, and what value they offer as input to the direction that you want to take your enterprise.

This chapter covers the following topic:

- Choosing the best IMS connectivity solution set

1.1 Choosing the best IMS connectivity solution set

Table 1-1 presents a high-level description of technologies and specific products that are available to support various options for IMS connectivity.

Here is some terminology that is related to this topic:

- ▶ **Java EE Connector Architecture (JCA)**

A Java based technology solution for connecting application servers and enterprise information systems (EIS) as part of an enterprise application integration solution. The JCA defines a standard set of system-level contracts between the Java EE application server and a resource adapter that manages the connection, transactions, security, work, lifecycle, transaction, and message inflow.

- ▶ **Java Message Service (JMS)**

An asynchronous-based messaging interface for exchanging of data between computers using messaging services in support of Java programs. JMS allows application components that are based on the Java Platform, Enterprise Edition to create, send, receive, and read messages that can be text strings, serialized Java objects, or XML documents. Messages can be exchanged using the point-to-point or the publish-and-subscribe model.

- ▶ **Web services and SOAP**

XML-based interfaces and protocols that allow heterogeneous applications to discover and communicate with each other in a platform and language independent way

- ▶ **IMS TMRA**

IMS TM Resource Adapter.

- ▶ **WebSphere MQ and IMS Bridge**

WebSphere Message Queue with the IMS Bridge facility.

- ▶ **WebSphere Process Server is replaced by the IBM Business Process Manager**

Table 1-1 Technologies and products that are associated with IMS connectivity solutions

Technology	Products that are involved in that technology	Architecture	Overview
IMS Enterprise Suite SOAP Gateway	IMS Connect	SOAP	SOAP Gateway is a lightweight web service server that has been customized to support IMS messaging. Details are available in Chapter 5, “SOAP application technology” on page 169.
IMS Enterprise Suite Connect APIs	IMS Connect	N/A	This is a programming interface to allow the customization of IMS Connect TCP/IP client applications.

Technology	Products that are involved in that technology	Architecture	Overview
WebSphere Application Server	IMS Connect and IMS TMRA.	JCA	This is a Java Platform, Enterprise Edition Connector Architecture (JCA) adapter solution that provides Java applications access to IMS with integrated connection pooling, transaction management with two-phase commit support, and levels of security management
	WebSphere Optimized Local Adapter (WOLA)	JCA	This is an optimized local IBM z/OS adapter with high speed cross memory mechanisms for transaction propagation between IMS and WebSphere Application Server z/OS on the same LPAR
WebSphere Process Server / IBM Business Process Manager	IMS Connect and IMS TMRA	JCA, SOAP	This facility provides business process choreography functionality
WebSphere Process Server / IBM Business Process Manager	WebSphere MQ and IMS Bridge	JMS, SOAP	
WebSphere Message Broker (WebSphere MB)	IMS Connect and IMS TMRA	JCA, SOAP	In a heterogeneous infrastructure environment, these solutions are best for the handling of both standard and nonstandard based applications, protocols and data formats.
	WebSphere MQ and IMS Bridge	JMS, SOAP	
	IMS SOAP Gateway		
WebSphere Transformation Extender	IMS Connect and IMS TMRA	JCA	Best suited for transformation and validation capabilities allowing aggregating of disparate and large volumes of data with one-pass lookup.
	WebSphere MQ and IMS Bridge	JMS, SOAP	
IBM DataPower	IMS Connect	SOAP	An application-based Enterprise Service Bus with pre-built software and hardware combinations that provide high-speed web service and XML transformations with advanced security and web service standardization that is built in.
	WebSphere MQ and IMS Bridge	JMS, SOAP	
IMS Data Provider for Microsoft.NET	A component of the IMS Enterprise Suite	.NET framework	Allows streamlined and standards-based way of accessing IMS databases from any .NET applications using SQL.
IBM Worklight®	HTTP SOAP, TCP/IP XML, and IMS Connect	Mobile device framework	Provides a platform to build, run, and manage rich, cross-platform mobile applications.

1.1.1 Considerations that are related to your choice of service

Both your existing server environment and access requirements must be considered to determine the best solution for you:

- ▶ If you require either basic web services or Java access to IMS

If you have an existing Java EE server or one of the IBM servers that needs web services or Java access to IMS, the JCA architecture with the IMS TM resource adapter is the best option for direct access to IMS. This option offers full quality of service (QoS) with no extra middleware required. If you have WebSphere Application Server on z/OS, then WOLA offers you optimized access.

- ▶ If you require middleware to ensure recoverable message deliveries

If you have an existing Java EE server or one of the IBM servers that needs web services or Java access to IMS, and you need a middleware to ensure message delivery with full message recovery capability, then JMS with WebSphere MQ is the best integration option.

- ▶ If you require platform independent loosely coupled interfaces

If you need to interact with business partners or applications in a platform independent and loosely coupled fashion, and IMS transactions must be directly accessible without going through or requiring other Java applications or additional programming, IMS Enterprise Suite SOAP Gateway is the best option for direct SOAP access to IMS.

IMS Data Provider for Microsoft .NET is the obvious choice when interfacing with any .NET application.

If you have DataPower, then take advantage of its support of IMS for a high-speed web service process with advanced security support.

- ▶ If you require simple APIs to access IMS data stores and application sets

If you have an existing in-house server that needs a simple API to access IMS, you can use a high-level API, such as the IMS Connect API or WebSphere MQ API, which offers the flexibility to integrate with your existing application interfaces.

- ▶ If you require mobile enhancement solutions

IBM Worklight includes a suite of integration adapters (HTTP for REST and SOAP support and SQL) to allow this platform to connect to “back-end” systems, such as IMS.



IMS Connect

IMS Connect is a communication gateway that operates in an address space on the z/OS platform between a service consumer (client) and IMS (if inbound), or IMS and a service producer (server) (if outbound). It interfaces with any TCP/IP supported environment, including Linux.

IMS Connect enables the following functions:

- ▶ Distributed clients that exchange messages with IMS Transaction Manager (TM) by using TCP/IP connections and Open Transaction Manager (OTMA).
- ▶ Local z/OS clients that use the local option with IMS TM Resource Adapter to exchange messages with IMS TM by using the z/OS Program Call (PC) function (the *local option*) and OTMA.
- ▶ Distributed clients that exchange messages with IMS DB by using TCP/IP connections and the Open Database Manager (ODBM) component of the IMS Common Service Layer (CSL). This facility is available as of IMS Version 11.
- ▶ Multiple Systems Coupling (MSC) users that send messages from one IMS system to another by using IMS-to-IMS TCP/IP connections. This facility is available as of IMS Version 12.
- ▶ Intersystem communication support (ISC) using TCP/IP between IBM CICS® and IMS.
- ▶ IMS Operators that use the IMS Control Center to issue commands to an IMSplex and receive command replies by using TCP/IP and the IMS Operations Manager (OM).

Performance measurements that were obtained at the Silicon Valley Laboratory demonstrated that 12,000 transactions per second can be processed with a single IMS Connect instance supporting a single IMS. This can be increased by using parallel IMS Connect instances. A four-way IMS Connect configuration in a single IMS Version 12 system was clocked around 25,000 transactions per second.

At the same IBM laboratory, we achieved a message sending rate of over 9,500 messages a second between a local ICON (IMS Connect) and remote ICON image over a Internet Protocol network. This test was performed on an IBM System z10® Enterprise Class Model E64.

For more information about this test, see the following website:

http://public.dhe.ibm.com/software/data/sw-library/ims/IMS_Version_12_Performance

This chapter covers the following topics:

- ▶ IMS Connect overview
- ▶ Commit processing message flows
- ▶ Installing IMS Connect
- ▶ IMS Connect implementation and configuration process
- ▶ IMS Connect configuration enhancements
- ▶ IMS Connect user exits
- ▶ IMS Connect user exit enhancements
- ▶ IMS Connect high availability facilities
- ▶ IMS Connect high availability enhancements
- ▶ IMS Connect operations and command support
- ▶ IMS Connect operations and command enhancements
- ▶ Accessibility through IMS Connect
- ▶ IMS Connect accessibility enhancements
- ▶ IMS Connect client programming interfaces
- ▶ IMS Connect Client programming interface enhancements
- ▶ IMS Connect security
- ▶ IMS Connect security enhancements
- ▶ IMS Connect scalability and performance
- ▶ IMS Connect scalability and performance enhancements
- ▶ IMS Connect diagnostic tests
- ▶ IMS Connect diagnostic enhancements

2.1 IMS Connect overview

As shown in Figure 2-1, a service consumer can connect through TCP/IP (local connect is also available) to the IMS server over the IMS Connect gateway. The local support facility allows communication by using Program Calls (PC) without requiring TCP/IP from a web serving application to IMS in a z/OS environment, which eases the management in this environment.

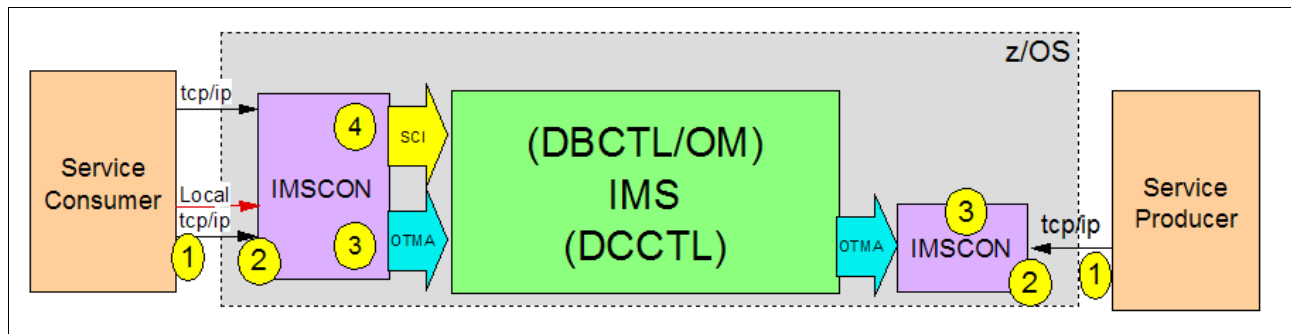


Figure 2-1 IMS Connect overview

Here is an explanation of the numbered items in Figure 2-1:

1. The local option is also available (in place of communication through TCP/IP).
2. From a TCP/IP point of view, IMS Connect is always a server.

3. A callin or callout function can be in the same IMS instance.
4. The Structured Call Interface (SCI) IMS address space is used for communication between IMS Connect and the Operations Manager address space as part of the distributed control center support. SCI is also used for ODBM, MSC, and ISC traffic.

Traditionally, IMS Connect is used as a gateway to reach the DCCTL (communication portion) of IMS. Since IMS Version 9, it is also used as a communication path for the IMS Operation Management (OM) component for distributed IMS single point of control (SPOC), as provided from a distributed control center to operate IMS within an IMSplex. The Structured Call Interface (SCI) component of IMS is used for the IMS Connect to Operations Manager (OM) communication.

IMS Connect interfaces with Open Transaction Manager Access (OTMA) to provide a direct communication path from clients to IMS applications. OTMA is a feature of IMS that enables any z/OS client (IMS Connect in this case) to access IMS applications through the cross-system coupling facility (XCF) services. For more information about OTMA, see Chapter 4, “Open Transaction Manager Access” on page 127.

Request messages that are received from TCP/IP clients, using TCP/IP connections or local option clients using the z/OS program call (PC), are passed to IMS through XCF. Then, IMS Connect receives response messages from IMS and passes them back to the originating TCP/IP or local option clients.

IMS Connect supports TCP/IP clients communicating with socket calls, but it can also support any TCP/IP client that communicates with a different input data stream format. User-written message exits can run in the IMS Connect address space to convert customer message format to OTMA message format before IMS Connect sends the message to IMS. The user-written message exits also convert OTMA message formats to customer message formats before sending a message back to IMS Connect. IMS Connect then sends the output to the client.

The IMS Connect configuration supports multiple IMS Connect connections accessing the same IMS system and a single IMS Connect connection accessing multiple IMS systems. If the data store goes down, the status of the data store is sent to IMS Connect from IMS OTMA through XCF. When the data store is brought back up and restarted, IMS Connect is notified and automatically reconnects to it. You do not need to manually reconnect to the data store.

If you stop and restart TCP/IP, as of IMS V11, IMS Connect automatically reconnects. However, the IMS Connect clients must reconnect.

IMS Connect was augmented in IMS V11 to be an Open Database Manager (ODBM) client, which allows distributed applications to use the TCP/IP protocol to communicate through IMS Connect to access any database in the entire IMSplex.

2.2 Commit processing message flows

The Commit and Synclevel parameters are important to understand because they influence both the integrity of transactions and performance.

- ▶ CM0: Commit 0 (commit then send) indicates that the sync point commit decision is taken by IMS before message shipment occurs. You could think of this as asynchronous processing.
- ▶ CM1: Commit 1 (send then commit) means that after processing the transaction, the response is sent right away to the client and the decision to commit is returned to IMS by an external element. This is synchronous in nature.

2.2.1 Commit-then-send (commit mode 0) flow

The commit-then-send flow is also known as the standard IMS flow because it is the way IMS has traditionally worked. The output message is enqueued to the output queue (in the case of OTMA, to the Tpipe structure) and sent after the transaction program has reached its commit point and completed, as shown in Figure 2-2.

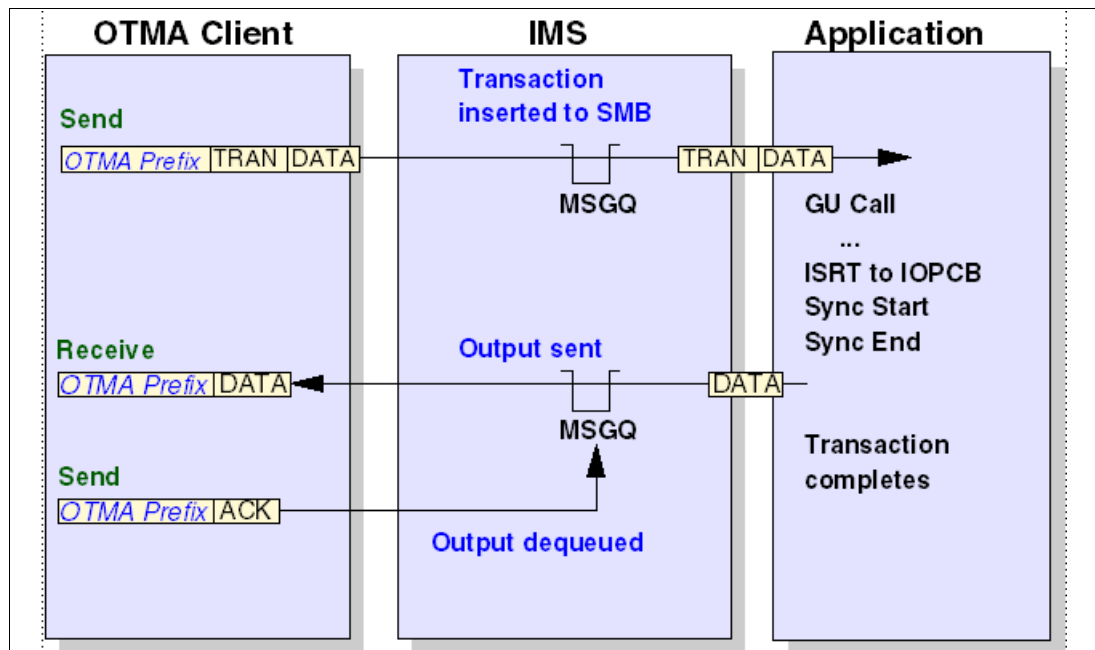


Figure 2-2 Commit-then-send message flows

To ensure that client transactions are processed and that they are processed only once, OTMA provides a protocol for synchronizing transactions. This is done by defining the Tpipe as synchronized and by using commit mode 0. IMS then waits for the response from the client (acknowledgment) before the output is dequeued from the message queue. This provides message recoverability similar to that provided for Set and Test Sequence Number (STSN) terminals, such as SLUP, 3600, and ISC terminals. If an ACK is not received or a NAK is sent by the client, the output message is not dequeued.

Message recoverability is applicable only for commit-then-send mode transactions. Any input messages on the message queue that are marked unrecoverable are discarded during an IMS restart. Any input messages on the message queue that are marked recoverable are left on the message queue and are eligible for scheduling after IMS restart completes.

2.2.2 Send-then-commit message (commit mode 1) flows

With send-then-commit mode, there are three different levels of synchronization that can be used: *none*, *confirm*, or *sync point*. The message flow is different in all these different cases. With commit-then-send mode, the synchronization level parameter is ignored, and the synchronization level of confirm is always used.

With the send-then-commit flow, the output is not enqueued; instead, it is sent directly to the client before the transaction is committed. Therefore, the output is unrecoverable if there is a transaction abort. After sync point is complete, OTMA sends another output indicating whether the sync point was successful or failed. This is called a *deallocation message*. If the sync point was successful, a deallocate confirmation message is sent, and the OTMA client knows that the transaction has not backed out. If the sync point was not successful (for example, DB2 voted “no” during two-phase commit), OTMA sends a deallocate abort message, and the OTMA client knows that the transaction has backed out. It then must decide what to do with the output message.

2.2.3 Send-then-commit (sync level=none)

With send-then-commit flow, IMS sends the output message, but does not request or wait for an acknowledgement from the client. After the message is sent, sync point processing continues. Even though the OTMA client gets the message, it should not process the message until the deallocate message is received from OTMA.

Figure 2-3 shows the message flow for send-then-commit with sync level=none and the deallocate confirmation message. If the sync point does not complete successfully, OTMA sends the deallocate abort instead of the deallocate confirmation message, and then the client discards the output message.

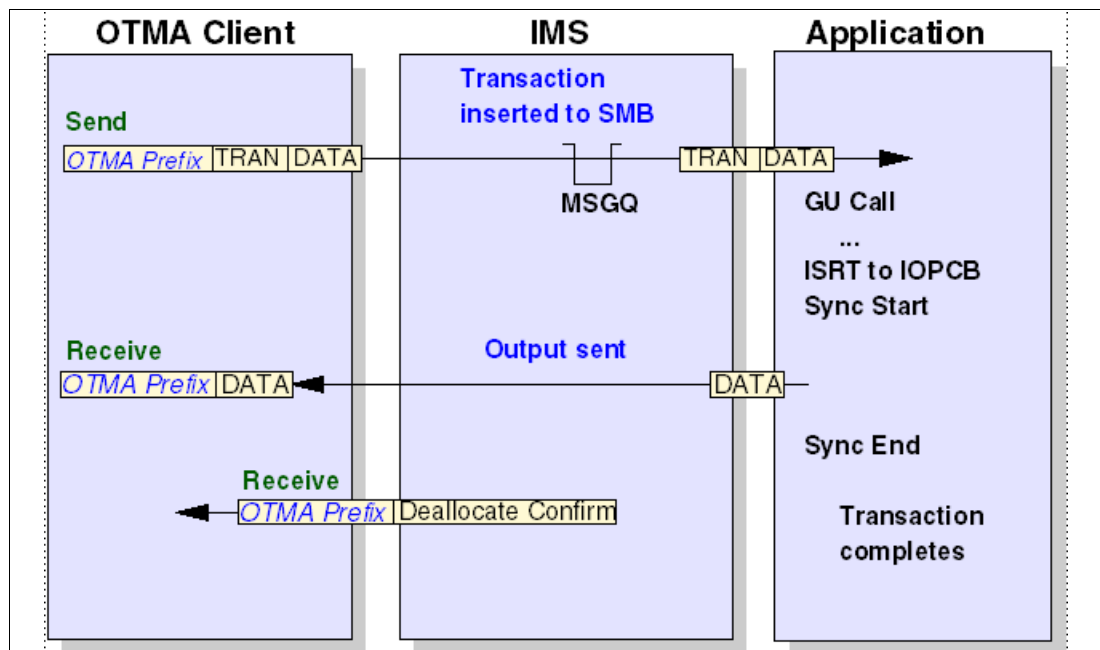


Figure 2-3 Send-then-commit message flow with sync level=none, deallocate confirm

2.2.4 Send-then-commit (sync level=confirm)

If the synchronization level is set to confirm in the state-data section, IMS sends the output message, request, and acknowledgement (ACK) or negative acknowledgement (NAK) before continuing sync point processing. This increases region occupancy.

If an ACK is returned, sync point processing continues. Even though the OTMA client has received the message and returned an ACK, it still does not process the message until the deallocate message is received.

Figure 2-4 shows the deallocate confirmed flow. Again, if the sync point does not complete successfully, OTMA sends the deallocate abort instead of the deallocate confirmation message, and then the client discards the output message.

If a NAK is returned, the transaction abends with code U0119. The database updates are backed out, and message DFS555I is sent to the client. The client then sends an ACK for the DFS555I message, and IMS responds with the commit aborted message.

While OTMA is waiting for an ACK/NAK, a `/DIS TMEBER xxx Tpipe yyy` command shows a status of WAIT_A (waiting for ACK). If you enter the `/STOP TMEBER xxx Tpipe yyy` command, a NAK is generated and the transaction abends with code U0119.

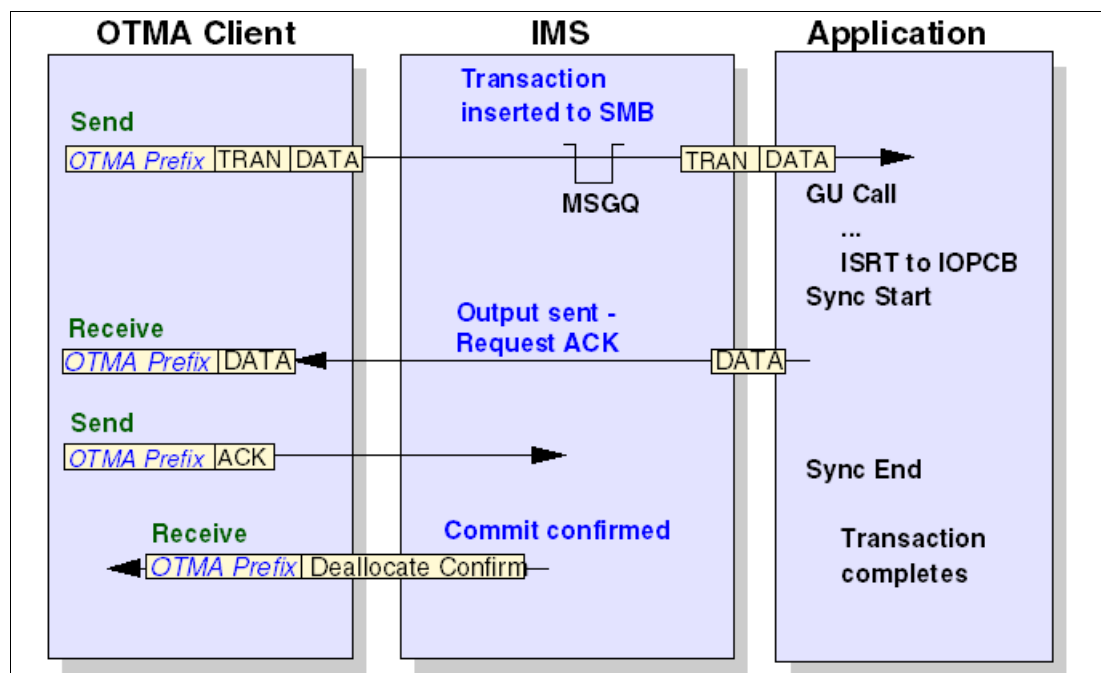


Figure 2-4 Send-then-commit message flow with sync level=confirm

2.2.5 Send-then-commit (sync level=sync point)

Figure 2-5 on page 13 shows the message conceptual flow in a situation where OTMA application programs and OTMA remote application programs participate with IMS in protected conversations with coordinated resource updates. z/OS Resource Recovery Services (RRS) does the global coordination of resources. RRS manages the sync point process on behalf of the conversation participants: the application program and IMS. IMS is acting as local resource manager. In IMS Version 13, you have the option of using XCF instead of RRS.

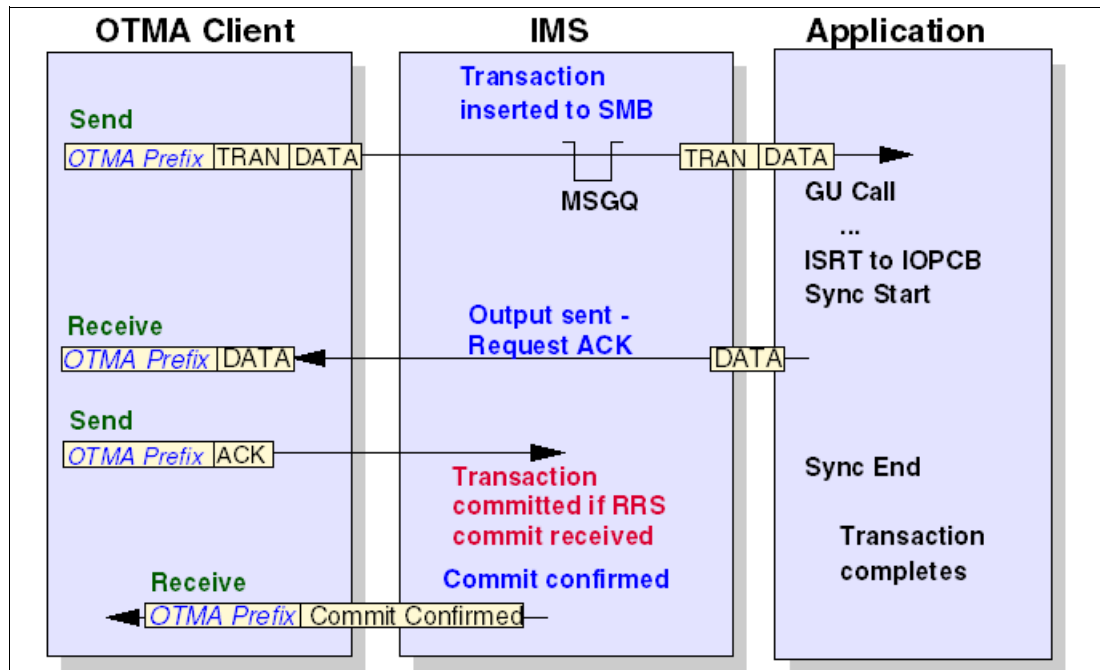


Figure 2-5 Send-then-commit message flow with sync level=sync point

In the resource coordination, two-phase commit (2PC) protocol is used. The two-phase commit protocol is a process involving the participants, the sync point manager, and the resource managers to ensure that, for updates that are made to a set of resources by the third-party application program, either all occur or none occurs. In simple terms, the application program decides to commit its changes to some resources. This commit is made to the sync point manager who then polls all of the resource managers as to the feasibility of the commit call. This is the preparation phase, often called phase 1.

Each resource manager votes yes or no, and when the sync point manager has gathered all the votes, phase 2 begins. If all votes are to commit the changes, the phase 2 action is commit. Otherwise, phase 2 becomes a backout. System failures, communication failures, resource manager failures, or application failures are not barriers to the completion of the two-phase commit process.

The work that is done by various resource managers is called a unit of recovery (UOR) and spans from one consistent point of the work to another consistent point, usually from one commit point to another. It is the unit of recovery that is the object of the two-phase commit process.

2.2.6 Persistent sockets

You can think of a non-persistent socket as one that allows input and then output to flow and then closes. A dedicated persistent socket remains dedicated to a particular user-specified clientid for commit-then-send (CM0) interactions until released by a client application's request.

Shareable persistent sockets, conversely, can be shared (serially reused) by multiple applications running either send-then-commit (CM1) or commit-then-send (CM0) interactions.

For performance reasons, it is important that the opening and closing of sockets are not repeated but managed by a “managed connection factory”, which allows for serial reuse of existing socket connections.

2.3 Installing IMS Connect

The following sections describe base functionality and then what was introduced in IMS V10, V11, V12, and V13. Although IMS Version 10 was withdrawn on 12 November 2012, you should be aware of the facilities that were introduced in IMS Version 10.

The IMS Connect installation is managed by normal SMP/E RECEIVE, APPLY, and ACCEPT services, and it is the same as other software products that run under a z/OS environment. The IMS Connect software distribution contains sample job control language (JCL) for the installation, which you can modify to meet the requirements of your environment. With IMS Version 9 onward, the IMS Connect feature is installed as a part of the IMS installation because it comes with the base IMS product.

2.4 IMS Connect implementation and configuration process

IMS Connect can be configured to meet your availability, capacity, security, and performance requirements. You can configure multiple IMS Connect images on multiple z/OS systems within a single Sysplex and distribute client requests to the IMS data stores.

IMS Connect runs as an z/OS job or started task and is controlled by two input files:

BPECFGxx	Defines the parameters for the Base Primitive Environment (BPE). These parameters are used to control common services, such as dispatching, waiting, and tracing.
HWSCFGxx	Specifies the environment for IMS Connect. This information is used to define the characteristics of the communication between IMS Connect and TCP/IP.

The TCP/IP client application requests a connection by specifying the host DNS name (resolves to the IP address of the target host), the IMS Connect port number, and the target IMS subsystem name. In an IMS Connect environment, this is called the data store ID, which identifies a specific statement in the IMS Connect configuration file, which directs the requests to a specific IMS system.

You can configure multiple IMS systems on multiple z/OS systems and distribute the client request to the data stores (IMSSs). To configure IMS Connect, complete the following steps:

1. Create an IMS Connect job or started task, as described in 2.4.1, “Creating an IMS Connect start procedure” on page 15.
2. Authorize the IMS Connect load library with the authorized program facility (APF), as described in 2.4.2, “Authorizing IMS Connect and BPE to the APF” on page 15.
3. Update the program properties table (PPT) in SYS1.PARMLIB. Updating the PPT allows IMS Connect to run in authorized supervisor state and in key 7. For more information, see 2.4.3, “Updating the program properties table” on page 16.
4. Enable Internet Protocol Version 6 (IPV6) for IMS Connect, as described in 2.4.4, “Enabling Internet Protocol Version 6 (IPV6) for IMS Connect” on page 16.

5. Create a BPE configuration member in support of IMS Connect, as described in 2.4.5, “Defining the IMS Connect Base Primitive Environment configuration” on page 17.
6. Create IMS Connect configuration members to hold the configuration statements that IMS Connect uses during initialization, as described in 2.4.7, “Defining a single IMS Connect configuration member” on page 28 onward.
7. Define IMS Connect security and (optionally) OTMA client security, as described in 2.4.10, “Defining IMS Connect security” on page 30.
8. Install the default user exits into the IMS Connect resident library, as described in 2.4.11, “Installing the default user exits into an IMS Connect resource library” on page 31.
9. Optionally, enable the IMS Connect XML message conversion support.

2.4.1 Creating an IMS Connect start procedure

Example 2-1 presents a sample deck that is associated with an IMS Connect image start procedure.

Example 2-1 IMS Connect startup JCL

```
//HWS PROC RGN=4096K,SOUT=A,
// BPECFG=BPEHWS2B,HWSCFG=HWC13B1
//*
//*****
//* BRING UP AN IMS CONNECT *
//*****
//STEP1 EXEC PGM=HWSHWS00,REGION=&RGN,TIME=1440,
//          PARM='BPECFG=&BPECFG,HWSCFG=&HWC13B1'
//STEPLIB DD DSN=IMS.SDFSRESL,DISP=SHR
//          DD DSN=CEE.SCEERUN,UNIT=SYSDA,DISP=SHR
//          DD DSN=SYS1.CSSLIB,UNIT=SYSDA,DISP=SHR
//          DD DSN=GSK.SGSKLOAD,UNIT=SYSDA,DISP=SHR
//PROCLIB DD DSN=USER.PROCLIB,DISP=SHR
//SYSPRINT DD SYSOUT=&SOUT
//SYSUDUMP DD SYSOUT=&SOUT
//HWSRCORD DD DSN=HWSRCORD,DISP=SHR
```

Note: IMS Connect requires the SYS1.CSSLIB and GSK.SGSKLOAD libraries (which are the C execution and z/OS system SSL libraries) only when SSL support is used. Many installations that use these libraries place them in the LINKLIST concatenation, in which case they are not needed in the JCL.

The HWSRCORD data set is the line trace data set. If you are not using BPE to manage the output of the IMS Connect Recorder Trace facility, you can enable and start a recorder trace that is managed by IMS Connect.

2.4.2 Authorizing IMS Connect and BPE to the APF

The SDFSRESL library in which the IMS Connect modules are in must be authorized to the APF. This can be done by editing the appropriate PROGxx member in the SYS1.PARMLIB and running the **SET PROG=xx** command. Coordinate with your z/OS systems programmer to ensure that these libraries are correctly authorized.

Example 2-2 provides a sample of establishing APF entries for IMS Connect.

Example 2-2 APF entries for IMS systems in SYS1.PARMLIB(PROGxx)

APF ADD	
DSNAME(IMS13Q.IMS13A.SDFSRESL)	VOLUME(SBOXI6)
APF ADD	
DSNAME(IMS13Q.IMS13B.SDFSRESL)	VOLUME(SBOXI6)
APF ADD	
DSNAME(IMS13Q.IMS13C.SDFSRESL)	VOLUME(SBOXI6)

2.4.3 Updating the program properties table

Because IMS Connect is ran in supervisor state and key 7, add an entry for it in the z/OS program properties table (PPT) by editing the SCHEDxx member of the SYS1.PARMLIB data set. The specification in the z/OS PPT must match the program specification in the EXEC statement of the IMS Connect startup JCL. You can specify either BPEINI00 or HWSHWS00. Example 2-3 presents this PPT entry.

Example 2-3 PPT entry that is required for IMS Connect

PPT PGMNAME(HWSHWS00)	/*PROGRAM NAME =HWSHWS00 */
CANCEL	/*PROGRAM CAN BE CANCELED */
KEY(7)	/*PROTECT KEY ASSIGNED IS 7 */
SWAP	/*PROGRAM IS SWAPPABLE */
NOPRIV	/*PROGRAM IS NOT PRIVILEGED */
DSI	/*REQUIRES DATA SET INTEGRITY */
PASS	/*CANNOT BYPASS PASSWORD PROTECTION */
SYST	/*PROGRAM IS A SYSTEM TASK */
AFF(NONE)	/*NO CPU AFFINITY */
NOPREF	/*NO PREFERRED STORAGE FRAMES */

Note: Use SWAP only if you are using only TCP/IP communications. If you are using the local option for client communications either by itself or with TCP/IP communications, you have to use NOSWAP.

To make the changes effective, perform either of the following actions:

- ▶ Perform an IPL of your z/OS system.
- ▶ Run the z/OS SET SCH= command.

2.4.4 Enabling Internet Protocol Version 6 (IPV6) for IMS Connect

To accomplish this task, complete the following steps:

1. Ensure that IMS Connect is running.
2. Customize the BPXPRMxx member:


```
FILESYSTYPE Type(INET) Entrypoint(EZBPFINI)
NETWORK DOMAINNAME(AF_INET)
DOMAINNUMBER(2)
MAXSOCKETS(2000)
TYPE(INET)
NETWORK DOMAINNAME(AF_INET6)
DOMAINNUMBER(19)
```

MAXSOCKETS(3000)
TYPE(INET)

BPXPRMxx contains the parameters that control the z/OS UNIX System Services (z/OS UNIX) environment and the file systems.

3. Recycle the TCP/IP stack. For more information about customizing this member, see *z/OS V1R12.0 UNIX System Services Planning*, GA22-7800-18.
4. Customize the IMS Connect configuration member by using the IPV6 parameter.
5. For each of the READ subroutines that are shown below that you use, determine whether the EXPREA_IPV6 bit is turned on in the EXPREA_FLAG2 field of the READ subroutine. If it is turned on, IPV6 is enabled.
 - HWSJAVA0
 - HWSSMPL0
 - HWSSMPL1
6. Map EXPREA_SOCKET6 to the AF_INET6 socket address structure.

2.4.5 Defining the IMS Connect Base Primitive Environment configuration

The IMS Connect address space is built on top of the IMS Connect Base Primitive Environment (BPE). Generally, you do not need to work with the IMS Connect BPE. However, you might need to change the default settings for certain IMS Connect BPE functions, such as storage management, internal tracing, dispatching, and other system service functions. IMS Connect supplies a configuration data set member for IMS Connect BPE system service functions that you can modify.

The IMS Connect BPE configuration parameter PROCLIB member defines the IMS Connect BPE execution environment settings for the IMS Connect address space. You specify the PROCLIB member name by coding BPECFG=*member name* on the EXEC PARM= statement in the IMS Connect address space startup JCL (see the IMS Connect startup JCL in Example 2-1 on page 15). Example 2-4 shows a sample IMS Connect BPE configuration file.

Example 2-4 Sample BPE configuration file

```
*****
* CONFIGURATION FILE FOR BPE WITH SOAP GATEWAY                                *
*****
LANG=ENU                                /* LANGUAGE FOR MESSAGES          */
                                         /* (ENU = US ENGLISH)            */

#
# DEFINITIONS FOR BPE SYSTEM TRACES
#
TRCLEV=(*,HIGH,BPE,PAGES=20)           /* DEFAULT TRACES TO HIGH        */
TRCLEV=(STG,MEDIUM,BPE)                /* STORAGE TRACE                  */
TRCLEV=(CBS,MEDIUM,BPE)                /* CONTROL BLK SRVCS TRACE        */
TRCLEV=(DISP,HIGH,BPE)                 /* DISPATCHER TRACE               */
TRCLEV=(AWE,HIGH,BPE)                  /* AWE SERVER TRACE               */
TRCLEV=(SSRV,HIGH,BPE)                 /* SYSTEM SERVICE TRACE           */

#
# DEFINITIONS FOR HTM TRACES
#
TRCLEV=(HACT,HIGH,HTM)                  /* HTM-ADM COMM TRACE             */
TRCLEV=(SHCT,HIGH,HTM)                  /* SERVER-HTM COMM TRACE          */

#
# DEFINITIONS FOR IMS CONNECT TRACES
```

```

#
TRCLEV=(*,HIGH,HWS,PAGES=20)          /* DEFAULT TRACES TO HIGH */
TRCLEV=(HWSI,HIGH,HWS,PAGES=100)       /* OTMA COMM ACTIVITY TRACE */
TRCLEV=(HWSN,HIGH,HWS,PAGES=100)       /* LOCAL OPT DRIVER ACTIVITY */
TRCLEV=(HWSW,HIGH,HWS,PAGES=100)       /* TCP/IP DRIVER ACTIVITY */
TRCLEV=(OTMA,HIGH,HWS,PAGES=100)       /* XCF CALLS TRACE */
TRCLEV=(TCPI,HIGH,HWS,PAGES=100)       /* TCP/IP CALLS TRACE */
TRCLEV=(RCTR,MEDIUM,HWS,EXTERNAL=YES)
#
# DEFINITION FOR NEW STYLE RECORDER TRACE
#
EXTTRACE(
  GDGDEF(
    DSN(IMS13Q.IMS13B.HWS.RCTR)
    UNIT(SYSALLDA)
    VOLSER(SBOXI6)
    SPACE(15)
    SPACEUNIT(TRK)
    BLKSIZE(32760)
  )
  COMP(HWS)
)
EXITMBR=(HWSEXITO,HWS)
# DEFINITIONS FOR SOAP GATEWAY

```

2.4.6 Base IMS Connect configuration statement parameters

You use the IMS Connect configuration member of the IMS PROCLIB data set, `HWSCFGxx`, to specify environmental settings for IMS Connect. The values for parameters in the IMS Connect configuration member define how IMS Connect communicates with TCP/IP, Open Transaction Manager (OTMA), IMSplexes, Open Database Manager (ODBM), Multiple Systems Coupling (MSC), security software, and other IMS Connect instances. As of IMS V13, the IMS Connect configuration member supports the following statements:

- ▶ **ADAPTER**
- ▶ **DATASTORE**
- ▶ **HWS**
- ▶ **IMSPLEX**
- ▶ **ISC**
- ▶ **MSC**
- ▶ **ODACCESS**
- ▶ **RMTCICS**
- ▶ **RMTIMSCON**
- ▶ **RUNOPTS**
- ▶ **TCPIP**

ODACCESS is explained in more detail in Chapter 9, “IMS Explorer and Open Database connectivity” on page 347. Details are in the information center. The following sections contain description of the main configuration statements.

ADAPTER

The **ADAPTER** statement defines the characteristics of adapters that are used to convert XML input messages to other application programming languages, such as COBOL or PL/I.

Here are the **ADAPTER** statement parameters:

XML	No or Yes. This is the option to enable the XML adapter. No is the default. The XML adapter is used to convert the user data in the response message into XML. IMS Connect then sends the output message to the IMS Enterprise Suite SOAP Gateway.
MAXCVRT	Specifies the maximum number of XML converters that this instance of IMS Connect can load concurrently. The minimum value is 100 and the maximum is 2000. The default value is 100. If more than 100 converters are included in the BPE exit list, IMS Connect loads and unloads them from memory as needed.
MAXLSSSZ	Specifies the maximum language structure segment size. This value is passed to the XML converter when it is called. Valid values are 5 - 32767. This parameter is optional and defaults to 32767.

DATASTORE

The **DATASTORE** statement specifies each data store with which the IMS Connect communicates through IMS OTMA. You can define multiple **DATASTORE** cards; each one is an OTMA client.

IMS Connect uses this information to translate the logical data store name that is passed by the TCP/IP client into the IMS XCF member name and thus the IMS control region.

Here are the **DATASTORE** statement keyword parameters:

ID	<p>The data store name. This consists of alphanumeric character data, begins with an alphabetic character, and has a length 1 - 8 bytes. This ID name cannot be the same as the <i>tmember</i> on the IMSPLEX statement.</p> <p>The IMS Connect client passes to IMS Connect a data store ID to identify the IMS to receive the message. The data store ID that is supplied by the client must match with a data store name that is defined to IMS Connect.</p> <p>The IMS Connect user exit can override the data store ID that is supplied by the client.</p>
GROUP	The XCF cross system coupling facility group name for the IMS OTMA. IMS Connect uses this value to join the appropriate XCF groups. This group name must match the XCF group name that you define to the GRNAME in the IMS startup JCL or DFSPBxxx member because IMS Connect and IMS must be in the same XCF group to communicate. Each IMS Connect can join any number of groups.
MEMBER	The XCF member name that identifies IMS Connect in the XCF group that is specified by the group parameter. This name is the XCF name that IMS uses to communicate with IMS Connect in that XCF group. This XCF member name for IMS Connect must be unique in the data store definitions for all data stores that are members of the same XCF group.

TMEMBER	The XCF member name for IMS that IMS Connect uses to communicate with an IMS in its XCF group. This target member name must match the member name IMS uses when it joins the XCF group. The XCF member name for IMS is specified in the IMS startup JCL by the OTMANM parameter in the IMS startup JCL or DFSPBxxx member. Each data store definition within an IMS Connect configuration member must contain a unique <i>tmember</i> name.
RRNAME	<p>The name of an alternative destination that is specified in a client reroute request. If this string is not provided, IMS Connect uses HWS\$DEF as the default name.</p> <p>It must be a string of 1 - 8 uppercase alphanumeric (A - Z, 0 - 9) or special characters (@, #, \$), left-aligned, and padded with blanks. IMS Connect translates lowercase characters to uppercase characters.</p> <p>The string is terminated by any blank or invalid character. The reroute name is truncated at any invalid character.</p>
APPL	<p>The TCP/IP APPL name that is defined to IBM RACF® in the PTKTDATA statement.</p> <p>This parameter is optional and defaults to blanks. If you are using PassTicket and user message exits, you must specify the APPL on the DATASTORE statement.</p>
DRU	<p>A 1 - 8 alphanumeric character field. The DRU keyword enables you to specify your own OTMA destination resolution user exit name that is passed to OTMA. The DRU exit is required to support asynchronous output to IMS Connect clients. The default is DFSYDRU0, but you can write your own exit.</p> <p>IMS Connect also provides the sample DRU exit routine (HWSYDRU0).</p>

HWS

The **HWS** statement defines characteristics that are specific to an instance of IMS Connect. It includes the following parameters:

ID	The IMS Connect name. It consists of alphanumeric character data, begins with an alphabetic character, and has a length of 1 - 8 characters.
RACF	<p>Y (yes), or N (no). Determines whether the password and user ID (provided by either the client application or a user exit routine) are passed to RACF for authentication. When RACF=Y, IMS Connect issues a RACROUTE REQUEST=VERIFY command to authenticate the user that is associated with incoming messages.</p> <p>This setting can also be changed by using the IMS Connect SETRACF command.</p> <p>If this is set to N, no RACF validation of the user ID is done, and the user ID (if provided) is simply passed to IMS. N is the default.</p> <p>Even if the value is Y, the IMS Connect user exit can set the trusted user flag and tell IMS Connect to not issue the RACF call.</p>
RRS	Y (yes), or N (no). Defines whether RRS should be enabled. This enables two-phase commit. N is the default.

XIBAREA You can use the exit interface block (XIB) and the user area it includes to store information that is used by your exit routines. This parameter specifies the number of fullwords that are allocated for the XIB user area. Both the user initialization exit routine (HWSUINIT) and the user message exit routines (HWSSMPL0 and HWSSMPL1) can access and modify the XIB user area.

The default value is 20; the maximum value is 500. If you do not specify a value for this parameter, or you specify a value outside of 20 - 500 range, the system uses the default value of 20.

IMSPLEX

The **IMSPLEX** statement specifies each IMS Operations Manager (OM) with which IMS Connect communicates through the IMS Structure Call Interface (SCI). You can have multiple **IMSPLEX** statements.

Here are the **IMSPLEX** statement keyword parameters:

MEMBER This name is passed to the SCI as the name of the IMS Connect that is communicating with the IMS OM through the SCI.

TMEMBER The name of the SCI to which IMS Connect communicates. The **tmember** name consists of alphanumeric character data, begins with an alphabetic character, and has a length of 1 - 5 bytes.

This name must be equal to the name specified in the SCI initialization PROCLIB member **IMSPLEX** in the parameter **NAME**.

The **TMEMBER** name cannot be the same name as the ID name on the **DATASTORE** statement.

ISC

The **ISC** statement, in combination with an **RMTCICS** statement and the **CICSPORT** keyword on the **TCPIP** statement, defines to IMS Connect an Intersystem Communication (ISC) link between a local IMS system and a remote IBM CICS Transaction Server for z/OS subsystem.

Here are the **ISC** statement parameters:

CICSAPPL For an ISC TCP/IP connection to a CICS subsystem, this value must match the value that is specified by CICS on the **APPLID=** parameter of the DFHSIT macro definition in the remote CICS subsystem.

CICSNETID For an ISC TCP/IP connection to a CICS subsystem, this NETWORK ID value is either the IBM VTAM® NETID or the **UOWNETQL=** parameter of the DFHSIT macro definition in the remote CICS subsystem.

The **CICSNETID** value must match the remote system NETWORK ID.

CICSPORT For ISC links to remote CICS subsystems, this is the local port that this ISC link uses. This value must match a port number that is defined on a **CICSPORT** parameter in the **TCPIP** configuration statement.

ID A unique ID for this **ISC** statement. The name must start with an alphabetic character and can be 1 - 8 alphanumeric characters in length.

IMSPLEX) Enables IMSplex communications between IMS Connect and ISC. IMSplex communications are managed by the Structured Call Interface (SCI) component of the IMS Common Service Layer (CSL).

MEMBER	A 1- to 8-character alphanumeric name that identifies IMS Connect in the IMSplex. The name must start with an alphabetic character. IMS Connect registers this name with SCI.
TMEMBER	The name of the IMSplex that IMS Connect is joining, as specified on the IMSPLEX (NAME=) statement of the CSLSLxxx member of the IMS PROCLIB data set of the SCI instance that is managing communications between IMS Connect and the IMSplex.
LCLIM	<p>This is the IMS ID of the local IMS system that is registered with SCI in the IMSplex. It is a 1- to 8-character alphanumeric name that begins with an alphabetic character.</p> <p>The values of the NODE keyword and the LCLIMS keyword must create a unique pair among all ISC statements in the IMS Connect configuration member.</p>
NODE	The name of this ISC node that is defined to the local IMS. For static terminal definitions, the value that is specified on the NODE parameter must match both the NAME parameter on the TERMINAL system definition macro and the nodename value of the ISCTCPIP parameter on the DFSDCxxx PROCLIB member. For dynamic terminal definitions, the value that is specified on the NODE parameter is used only in the /OPNDST command when a user initiates a new session with the CICS subsystem that is specified on the CICSAPPL parameter.
RMTCICS	For a connection to a CICS subsystem, this parameter specifies the ID of the RMTCICS statement that defines the TCP/IP connection that this ISC link uses to communicate with a remote CICS subsystem. This value must match the value of the ID= parameter of an RMTCICS statement that is defined in the IMS Connect configuration PROCLIB member.

MSC

The **MSC** statement defines a one-way send path for an MSC physical link between a local IMS system to a remote IMS system. The physical link send path uses a TCP/IP socket connection that is established by this IMS Connect instance with a remote IMS Connect instance. The connection that is used by the physical link must be defined to this IMS Connect instance by an **RMTIMSCON** statement.

Here are the **MSC** statement parameters:

GENIMSID	<p>This is an optional parameter. It specifies the MSC TCP/IP generic IMS ID for each IMS system participating in the local IMSplex. The same ID is specified in the GENIMSID parameter of the DFSDCxxx member of the IMS PROCLIB data set.</p> <p>The GENIMSID parameter accepts a 1 - 8 character alphanumeric name that begins with an alphabetic character. The value of GENIMSID cannot be the same name as the values that are specified on either the LCLIMS or RMTIMS parameters.</p>
IMSPLEX=()	Enables IMSplex communications between IMS Connect and MSC. IMSplex communications are managed by the Structured Call Interface (SCI) component of the IMS Common Service Layer (CSL).

MEMBER	A 1- to 8-character alphanumeric name that identifies IMS Connect in the IMSplex. IMS Connect registers this name with SCI. For MSC communications, this name must match the name that is specified on the LCLICON parameter of the MSPLINK macro definition of the local IMS system.
TMEMBER	The name of the IMSplex that IMS Connect is joining, as specified on the IMSPLEX (NAME=) statement of the CSLSIxxx member of the IMS PROCLIB data set of the SCI instance that is managing communications between IMS Connect and the IMSplex.
LCLIMS	For a link to a non-XRF IMS system, this parameter specifies the IMS ID of the local IMS system that is registered with SCI in the IMSplex. It is a 1- to 8-character alphanumeric name that begins with an alphabetic character.
LCLPLKID	<p>The local name of the MSC physical link. The name that is specified on the LCLPLKID parameter identifies the MSC physical link to IMS Connect. The name also associates the definitions in this MSC statement with the physical link definitions in an MSPLINK macro statement on the local IMS system.</p> <p>This name must match the name that is specified on the LCLPLKID parameter of the MSPLINK macro.</p>
RMTIMS	This is the IMS ID of the remote (target) IMS system that is registered with SCI in the remote IMSplex. It is a 1- to 8-character alphanumeric name that begins with an alphabetic character.
RMTIMSCON	The remote IMS Connect connection to use for MSC messages. The value of RMTIMSCON must match the value of the ID parameter of one of the RMTIMSCON statements that are specified in the local IMS Connect configuration.
RMTPLKID	The remote name of the MSC physical link that is specified on the LCLPLKID parameters of both the MSC statement of the remote IMS Connect and the MSPLINK macro of the remote IMS system. The name must start with an alphabetic character and can be 1 - 8 alphanumeric characters in length.

ODACCESS

The **ODACCESS** statement defines connection attributes for IMS DB communications. On the client side, the **ODACCESS** statement defines connection attributes between IMS Connect and the IMS Universal drivers and other IBM DRDA® clients. On the server side, the **ODACCESS** statement defines attributes for connections between IMS Connect and the CSL Open Database Manager (ODBM). IMS Connect must register with ODBM to enable access to IMS DB for clients that use the IMS Open Database architecture.

Here are the **ODACCESS** statement parameters:

DRDAPORT	<p>Defines the port numbers, the TCP/IP keep alive value, and the timeout values for the ports that IMS Connect uses to provide access to IMS for client applications of the Open Database APIs and user-written DRDA client applications.</p> <p>You can define up to 50 TCP/IP ports for an instance of IMS Connect. The total combined number of ports that are defined on all parameters in an IMS Connect configuration member cannot exceed 50 ports.</p>
-----------------	---

Here are the subparameters of **DRDAPORT**:

ID	The port number of the port that is defined by the DRDAPORT parameter.
KEEPAV	A 1- to 8-character decimal field that sets the interval after which the keepalive mechanism of the z/OS TCP/IP layer sends a packet on idle connections on this port to maintain the connections.
PORTTMOT	Defines the amount of time that IMS Connect waits for the next input message from a client application that is connected on a DRDA port before IMS Connect disconnects the client.
IMSPLEX= ()	Enables IMSplex communications between IMS Connect and ODBM. IMSplex communications are managed by the Structured Call Interface (SCI) component of the IMS Common Service Layer (CSL).
ODBAUTOCONN	Specifies whether IMS Connect automatically connects to new and existing instances of ODBM within an IMSplex.
ODBMTMOT	Defines the amount of time that IMS Connect waits for both of the following conditions: <ul style="list-style-type: none">– A response message on connections with ODBM.– An initial input message after a socket connection is established on connections with a client application

RMTCICS

The **RMTCICS** statement defines a TCP/IP connection to a remote IBM CICS Transaction Server for z/OS subsystem for ISC communications.

You can define multiple connections to one or more remote CICS subsystems by specifying a separate **RMTCICS** statement for each connection. Specifying multiple connections to the same CICS subsystem can, depending on the resources that are available at your installation, improve the performance of communications from IMS to the CICS subsystem.

Here are the **RMTCICS** statement parameters:

HOSTNAME	The host name of the remote CICS subsystem to which you are connecting.
ID	<p>The 1- to 8-character alphanumeric name that identifies this definition of a connection to a remote CICS subsystem.</p> <p>The value that is specified here must also be specified on the RMTCICS parameter of a corresponding ISC configuration statement in the HWSCFGxx member of the IMS PROCLIB data set.</p>
PORT	<p>The port number of the remote port that is used by the remote CICS subsystem. The port number must be specified as a 1- to 5-character decimal number.</p> <p>This port number must match the port number that is defined in the TCPIP SERVICE resource definition for this ISC link in the remote CICS subsystem.</p>

RMTIMSCON

The **RMTIMSCON** statement defines a TCP/IP connection to a remote IMS Connect instance. You can define multiple connections to one or more remote IMS Connect instances by specifying a separate **RMTIMSCON** statement for each connection.

Here are the **RMTIMSCON** statement parameters:

APPL	Specifies the 1- to 8-character alphanumeric application name to use in a RACF PassTicket that is sent to the remote IMS Connect instance
AUTOCONN	For OTMA connections, determines whether this IMS Connect instance connects to the remote IMS Connect instance during startup.
HOSTNAME	The host name of the remote IMS Connect instance that you are connecting to.
ID	The 1- to 8-character alphanumeric name that identifies this definition of a connection to a remote IMS Connect instance.
IDLETO	Specifies the amount of time that open socket connections can remain idle before they are terminated because of inactivity. The timeout interval is in hundredths of seconds.
IPADDR	The IP address of the remote IMS Connect instance to which you are connecting.
PERSISTENT	Defines the sockets that are used for this connection as persistent.
PORT	The 1- to 5-character decimal port number of the remote IMS Connect instance that you are connecting to. This port number must match a port number that is defined on either the PORT or PORTID parameter of the TCPIP configuration statement of the remote IMS Connect instance.
RESVSOC	The number of send sockets that IMS Connect reserves for use by this connection.
USERID	Specifies the 1- to 8-character alphanumeric user ID to use in a RACF PassTicket that is sent to the remote IMS Connect instance.

RUNOPTS

RUNOPTS is a 1 - 255 character string field that specifies the IBM Language Environment® runtime options to be used to override the IMS Connect default runtime options in support of SSL.

The **RUNOPTS** statement consists of only one parameter: **RUNOPTS**. This parameter is optional. IMS Connect passes the default values **POSIX(ON)** and **TRAP(OFF,NOSPIE)**, unless they are overridden by the **RUNOPTS** parameter.

TCPIP

The **TCPIP** statement defines IMS Connect communication with one TCP/IP.

Here are the **TCPIP** statement keyword parameters:

CICSPORT	Specifies a local TCP/IP port through which IMS Connect communicates with CICS.
ECB	Y (yes) or N (no). Specifies whether to use the TCP/IP exit or event control block (ECB) processing. ECB processing enhances IMS Connect performance by increasing throughput.

When ECB=N is specified (or left blank), IMS Connect runs with TCP/IP driving an IMS Connect exit.

When ECB=Y is specified, IMS Connect runs with TCP/IP driving IMS Connect with the posting of an ECB.

Set ECB=Y for the best performance.

EXIT

The name of the TCP/IP user message exit that receives control for messages that are received from and sent to TCP/IP clients. More than one exit can be defined as

EXIT=(EZAEXIT,EZBEXIT,EZCEXIT,HWSCSL00,HWSCSL01), for example, to a maximum of 254.

This parameter includes the IMS Connect sample exits and the user-written exits.

These user message exits support users other than IMS Connector for Java for OTMA linkage through IMS Connect to IMS. You do not have to include HWSJAVA0 in the EXIT= list. IMS Connect automatically loads the HWSJAVA0 exit, which is shipped with IMS Connect to enable IMS to support the IMS Connector for Java applications.

The user message exits for IMSplex support are HWSCSL00 and HWSCSL01 and must be specified here to ensure the activation of the IMS Control Center.

HWSUINIT is an IMS Connect exit, but is not a user message exit; you must not add it to the EXIT= parameter. If you add HWSUINIT to the EXIT= parameter, IMS Connect abends.

HOSTNAME

The name of the TCP/IP host. This is usually the jobname of the TCP/IP z/OS address space.

IPV6

Y (yes) or N (no). At IMS Connect start time, this parameter determines whether Internet Protocol Version 6 (IPV6) is enabled.

When N is specified or defaulted, IPV4 is used.

MAXSIZE

Specifies the maximum message size that is allowed in the 4-byte length field that precedes the IMS request message (IRM). Use the MAXSIZE= keyword to override the internal default of 10,000,000 bytes.

MAXSOC

A decimal field that is set to the maximum number of sockets per IMS Connect that an IMS Connect can open. This value must be a number 50 - 65535. The default value is 50.

The value that is specified results in one socket that is dedicated to a listen state per port and the remainder is available for connections. Therefore, if you specify 80 and have five ports, 75 physical connections can be made. The other five are used for the listen state sockets.

Now, you must consider that there are two parameters in the BPXPRMxx member of the SYS1.PROCLIB that are related to the allowed number of sockets. BPXPRMxx contains the parameters that control the z/OS UNIX System Services (z/OS UNIX) environment and the file systems. MAXSOCKETS sets a limit for the total number of sockets in a system, and MAXFILEPROC sets the maximum number of sockets for any job.

NODELAY

Specifies whether (Y) or not (N) the TCP/IP protocol option TCP_NODELAY is used. The default is no.

PORTAFF	Specifies that commit-then-send (CM0) output messages that IMS sends to this IMS Connect have affinity to the port on which IMS Connect received the original input message.
PORTID	<p>A 1- to 8-character field to define the TCP/IP ports, or a 5-character field with the value of LOCAL to define the local option connection. For TCP/IP port communications, it specifies the port number or numbers that binds to the socket (port for IMS Connect on which to listen).</p> <p>You can define more than one port as PORTID=(9999,8888,7777) to a maximum of 50. Port numbers must be 1 - 65535 and must be selected so that they do not conflict with other ports in the TCP/IP domain.</p>
RACFID	The default RACF ID for exits. Exits pass this ID to OTMA for security checking if the RACFID is not explicitly set in the incoming message or by the user exit.
SSLENVAR	The member name of the Secure Sockets Layer SLL initialization file.
SSLPORT	<p>A 1 - 5 numeric character decimal field to define SSL ports. For SSL port communication, it specifies the port number that binds to the socket (port for IMS Connect on which to listen with SSL).</p> <p>You can define up to 50 ports or an instance of IMS Connect, which must be numbered 1 - 65535. These ports must not conflict with any other ports that are selected in the TCP/IP domain or those selected under the PORTID parameter as basic TCP/IP ports.</p>
TCPIPQ	Specifies the number of connection requests without assigned sockets that IMS Connect can maintain. The minimum value is 50, which is also the default. The maximum is 2147483647.
TIMEOUT	<p>Time interval in hundredths of seconds after which IMS Connect disconnects the client if there is no response from IMS. The maximum value of the timeout is 2147483647 (X'7FFFFFFF') and the default is 0 (which means no timeout).</p> <p>IMS Connect uses the timeout value to determine the amount of time to wait for a response from IMS that is being sent to the client.</p> <p>IMS Connect also uses this timeout to disconnect a client socket connection that does not send messages. This timeout value on an IMS Connect read of the client applies only to the wait time between the socket connection and the first input from the client application. The timeout function is not activated between reads but only between the connection and the first IMS Connect read of the client application input.</p>
WARNINC	Specifies a warning level incremental percentage. After the WARNSOC value is reached, every time the number of sockets increases by the percentage value that is specified on WARNINC , IMS Connect reissues warning message HWSS0772W.
WARNSOC	Specifies a warning level when the number of sockets increases to a certain percentage of the MAXSOC limit.

2.4.7 Defining a single IMS Connect configuration member

You must create a configuration member in your PROCLIB data set to specify the IMS Connect environment. IMS Connect uses the information that it retrieves from the member to establish communication with IMS and TCP/IP. You can define several configuration members in the partition data set (PDS) to select from during the IMS Connect start. Specify the member name to be used in the **HWSCFG=** parameter of the IMS Connect startup JCL.

In the configuration member that is illustrated in Figure 2-6, and detailed in Example 2-5, IMS Connect is configured to include the ports that are defined for TCP/IP communications and the IMS OTMA group and member names for communication with IMS.

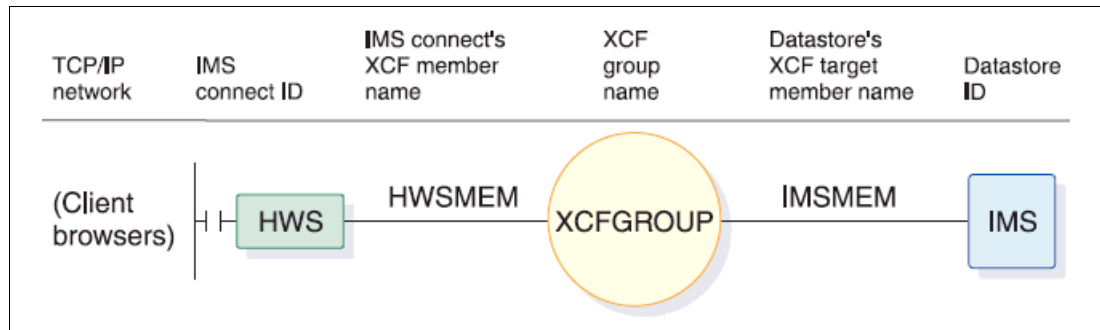


Figure 2-6 Simple IMS Connect configuration

- ▶ The IMS Connect ID is defined as HWS.
- ▶ The TCP/IP configuration defines the HOSTNAME as MVSTCPIP, the RACFID as RACFID, the PORTID as 9999, and the EXIT as HWSSMPL0.
- ▶ The data store configuration defines the ID as IMS, the GROUP as XCFGROUP, the MEMBER as HWSMEM, and the T MEMBER as IMSMEM.

Example 2-5 Sample configuration file

```
HWS=(ID=HWS,XIBAREA=100,RACF=N,RRS=Y)
TCPIP=(HOSTNAME=MVSTCPIP,PORTID=(9999),RACFID=RACFID,TIMEOUT=5000,EXIT=HWSSMPL0)
DATASTORE=(GROUP=XCFGROUP,ID=IMS,MEMBER=HWSMEM,DRU=HWSYDRUO,
T MEMBER=IMSMEM)
```

2.4.8 Defining multiple IMS Connect configuration members

In the example that is shown in Figure 2-7 on page 29, three IMS Connects are configured. Each IMS Connect has its own configuration member. Each IMS Connect uses a different port number for TCP/IP communications and can belong to multiple z/OS cross-system coupling facility (XCF) groups.

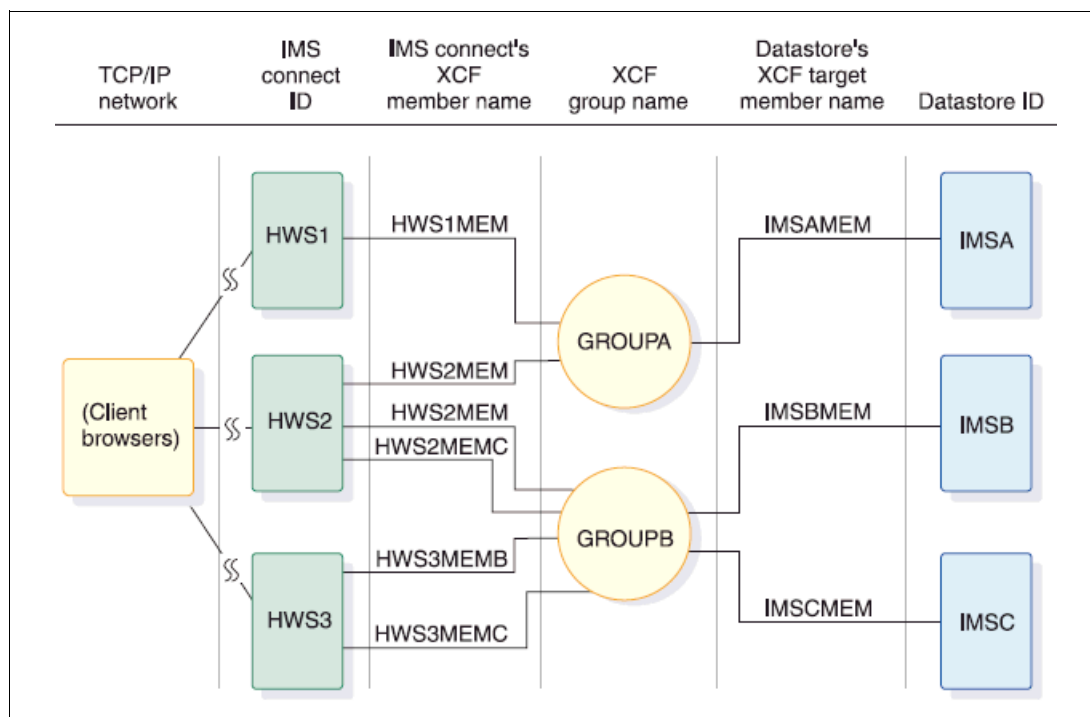


Figure 2-7 Configuration for multiple IMS Connects servicing multiple IMS images

Example 2-6 contains the configuration members for the three IMS Connect instances and their interfaces to the IMS data store members.

Example 2-6 IMS Connect configuration for three IMS Connect instances

```

*****
* IMS Connect example configuration member for HWS1
*****
HWS (ID=HWS1,RACF=N,XIBAREA=20)
TCP/IP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9999),MAXSOC=2000,TIMEOUT=8888,
EXIT=(HWSSMPLO))
DATASTORE (ID=IMSA,GROUP=GROUPA,MEMBER=HWS1MEM,TMEMBER=IMSAMEM,DRU=HWSYDRUO)
*****
* IMS Connect example configuration member for HWS2
*****
HWS (ID=HWS2,RACF=N,XIBAREA=20)
TCP/IP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9998),MAXSOC=2000,TIMEOUT=8888,
EXIT=(HWSSMPLO))
DATASTORE (ID=IMSA,GROUP=GROUPA,MEMBER=HWS2MEM,TMEMBER=IMSAMEM,DRU=HWSYDRUO)
DATASTORE (ID=IMSB,GROUP=GROUPB,MEMBER=HWS2MEM,TMEMBER=IMSBMEM,DRU=HWSYDRUO)
DATASTORE (ID=IMSC,GROUP=GROUPB,MEMBER=HWS2MEMC,TMEMBER=IMSCMEM,DRU=HWSYDRUO)
*****
* IMS Connect example configuration member for HWS3
*****
HWS (ID=HWS3,RACF=Y,XIBAREA=20)
TCP/IP (HOSTNAME=MVSTCPIP,RACFID=RACFID,PORTID=(9997),MAXSOC=2000,TIMEOUT=8888,
EXIT=(HWSSMPLO))
DATASTORE (ID=IMSB,GROUP=GROUPB,MEMBER=HWS3MEMB,TMEMBER=IMSBMEM,DRU=HWSYDRUO)
DATASTORE (ID=IMSC,GROUP=GROUPB,MEMBER=HWS3MEMC,TMEMBER=IMSCMEM,DRU=HWSYDRUO)

```

2.4.9 IMS Connect configuration statement for IMS DB support

IMS Open Database, which was introduced in IMS Version 11, provides distributed access to IMS database resources. It also helps drive open standards and open technology in IMS. The open standards that are introduced into this solution include the Java EE Connector Architecture, JDBC, and DRDA.

Figure 2-8 illustrates a simple configuration member for an IMS Connect instance that supports clients that connect to IMS DB in a DBCTL system.

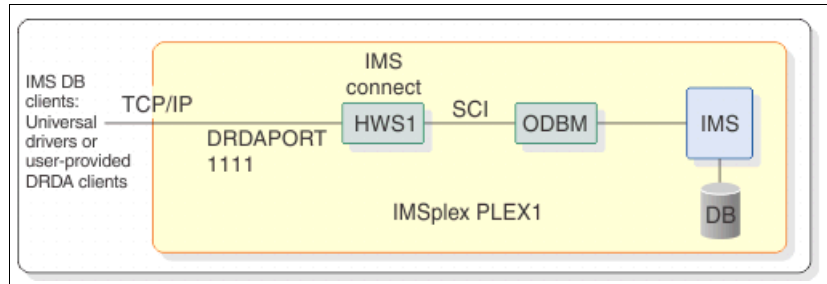


Figure 2-8 Simple IMS Connect system configuration for IMS DB support

Example 2-7 contains the definitions that are associated with Figure 2-8.

Example 2-7 Sample IMS Connect configuration definitions for IMS DB support

```
*****
* IMS Connect example for IMS Universal drivers and DRDA client support
*****
HWS (ID=HWS1,RACF=N,XIBAREA=20)
TCP/IP (HOSTNAME=MVSTCPIP,RACFID=RACFID,MAXSOC=2000)
ODACCESS (DRDAPORT=(ID=1111,KEEPAV=5,PORTTMOT=50),
IMSPLEX=(MEMBER=HWS1,TMEMBER=PLEX1),
ODBAUTOCONN=Y)
```

For more information about IMS Open Database, see Chapter 9, “IMS Explorer and Open Database connectivity” on page 347.

2.4.10 Defining IMS Connect security

If IMS is RACF-protected, you must start IMS Connect as a job with the JOB card specifying a valid *user ID* to make the connection from IMS Connect to IMS, or you can use the RACF started procedure table.

The user ID must have READ access to an IMSXCF.group.member. IMS OTMA provides security for the IMS XCF connection by defining and permitting IMSXCF.group.member in the RACF FACILITY class.

2.4.11 Installing the default user exits into an IMS Connect resource library

You must install the following two exits into your IMS Connect resource library (SDFSRESL) regardless of whether you intend to customize them because IMS Connect automatically loads these exits when it runs:

HWSJAVA0	User message exit for IMS Connector for Java clients, such as the IMS TM Resource Adapter
HWSUINIT	User initialization exit

You must compile and bind these exits before you run IMS Connect; otherwise, IMS Connect does not run. If you do not need to customize either of these two exits, you do not need to do anything else with them. Example 2-8 shows a sample JCL to bind the HWSUINIT exit.

Example 2-8 Sample JCL to install the default user exits

```
//HWSINIT JOB (ACTINF01),PGMRNAME,
// CLASS=A,MSGCLASS=Z,MSGLEVEL=(1,1),REGION=4M
//UNIT1 EXEC PGM=ASMA90,REGION=32M,
// PARM='DECK,NOOBJECT,SIZE(MAX,ABOVE)'
//SYSLIB DD DSN=IMS13Q.SDFSMA,DISP=SHR
// DD DSN=SYS1.MODGEN,DISP=SHR
// DD DSN=SYS1.MACLIB,DISP=SHR
//SYSPUNCH DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
// DSN=&&TEXT(HWSUINIT)
//SYSPRINT DD SYSOUT=*,
// DCB=(BLKSIZE=605),
// SPACE=(605,(100,50),RLSE,,ROUND)
//SYSUT1 DD UNIT=SYSDA,DISP=(,DELETE),
// DCB=BLKSIZE=13024,
// SPACE=(CYL,(16,15))
//SYSIN DD DSN=IMS13Q.SDFSMPPL(HWSUINIT),DISP=SHR
//UNIT2 EXEC PGM=IEWL,
// PARM=SIZE=(880K,64K),RENT,REFR,NCAL,LET,XREF,LIST,TEST
//SYSPRINT DD SYSOUT=A
//SYSLMOD DD DSN=IMS13Q.SDFSRESL,DISP=SHR
//SYSUT1 DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//TEXT DD UNIT=SYSVIO,DISP=(OLD,DELETE),DSN=&&TEXT
//SYSLIN DD *
INCLUDE TEXT(HWSUINIT)
ENTRY HWSUINIT
NAME HWSUINIT(R)
//
```

2.5 IMS Connect configuration enhancements

IMS Version 9 provides an integrated IMS Connect function, which is robust and functionally rich. More improvements were introduced with IMS Version 10 onward. For more information, see the following IBM Redbooks publications:

- ▶ *IMS Version 10 Implementation Guide A Technical Overview*, SG24-7526
- ▶ *IMS Version 11 Technical Overview*, SG24-7807
- ▶ *IBM IMS Version 12 Technical Overview*, SG24-7972

2.5.1 IMS Version 10: ACEE aging value support

The Access Control Environment Element (ACEE) is a control block that represents a verified user ID to IMS. The ACEE is used to determine the user's authorization to the IMS command or IMS transaction that is requested in the input message. When built in to OTMA, the ACEE for each user ID is cached and the aging value that is associated with each OTMA client, for example, IMS Connect, is kept in a table. The aging value is then used to determine when the cached control block expires and is refreshed.

IMS re-creates the ACEE if a message that is associated with the user ID is received but the age of the current ACEE is greater than the aging value. The aging value is used to balance performance (possible RACF I/O to refresh the ACEE) and integrity. For IMS Connect, the ACEE expiration value is specified during the client-bid process and is set to a default of no expiration.

Parameter **OAAV=**, in the **DATASTORE** statement, specifies the OTMA ACEE aging value. You can specify a value 300 - 999999 seconds. The default is 999999 seconds. OTMA requires a value of at least 300 to enable ACEE refreshes.

The **VIEWHWS**, **VIEWDS**, **QUERY MEMBER**, and **QUERY DATASTORE** commands output displays are enhanced to show the aging value that is in effect for the associated environment. Example 2-9 presents the output of the **VIEWHWS** command.

Example 2-9 VIEWHWS showing the ACEE aging value

```
R 218,VIEWHWS
  HWS ID=HWSI13B1 RACF=N  PSWDMC=R
    UIDCACHE=Y  UIDAGE=500
    MAXSOC=50  TIMEOUT=0  TCPIPQ=50
    NUMSOC=5    WARNSOC=80%  WARNINC=5%
    RRS=Y  STATUS=ACTIVE
    VERSION=V13 IP-ADDRESS=009.012.006.009
    .....
    TARGET MEMBER=I13BOTMA          STATE=N/A
    DEFAULT REROUTE NAME=HWS$DEF
    RACF APPL NAME=
    OTMA ACEE AGING VALUE=999999
    OTMA ACK TIMEOUT VALUE=120
    OTMA MAX INPUT MESSAGE=5000
```

Value: To ensure that the security information in the ACEE is correct, OTMA periodically refreshes the ACEE with the security information that is stored in RACF. It is by using the OAAV value that you can balance performance with in-core ACEEs with integrity.

2.5.2 IMS Version 10: Send-then-commit (CM1) ACK timeout control

IMS Connect supports the new OTMA timeout control function for send-then-commit CM1 synchronous interactions. OTMA provides a default value of 120 seconds, after which transactions that are held in *Wait-Syncpoint* or *Wait-RRS* status are released and backed out. If provided, the value in the **ACKTO** parameter of the IMS Connect configuration **DATASTORE** statement is passed to OTMA during client-bid processing.

A specified value of 0 is reset to 120. If the timeout value is zero or not specified, a value of 120 seconds as an OTMA ACK timeout default value is set. If the specified value is less than zero or greater than 255, an abend U3401 is issued. The value cannot be greater than what is defined in OTMA. If the ACKTO value is equal to or greater than the timeout that is specified in IMS by an OTMA descriptor or **/STA TMEMBER** command, OTMA ignores the IMS Connect request. However, if the ACKTO value is less than the current timeout value set by the OTMA descriptor or command, the value that is passed to IMS by the IMS Connect client-bid process is used for the timeout action for this member.

IMS Connect cannot disable the OTMA ACK timeout support. Disabling the capability can be done only in OTMA. Also, IMS Connect command output is enhanced to display the CM1 timeout value. The applicable commands include the **VIEWHWS** and **VIEWDS** output commands, and the **MVS MODIFY** command for **QUERY MEMBER** and **QUERY DATASTORE**.

Value: Until an ACK or NAK response is received by IMS from CM1 interactions, the IMS dependent regions remains occupied and continues to hold the necessary database locks. The use of the ACKTO parameter releases these resources when the timeout period is reached.

2.5.3 IMS Version 10: Message flood control

IMS Connect also takes advantage of the OTMA Message Flood Control capability to automatically monitor the growth of active input messages. OTMA provides a default value of 5000 messages, after which input messages from a specific IMS Connect instance are rejected.

If provided, an override value in the **MAXI** parameter of the IMS Connect configuration **DATASTORE** statement is passed to OTMA during client-bid processing.

Note the following considerations:

- ▶ A specified value of 0 is reset to 5000.
- ▶ A value 0 - 200 is reset to 200.
- ▶ Any value that is specified in error 9999 - 65535 is reset to 9999.
- ▶ Any value outside 0 - 65535 results in an abend U3401.
- ▶ The value cannot be greater than what is defined in OTMA. If the **MAXI** value is equal to or greater than the **INPUT** value specified in IMS by an OTMA descriptor or **/STA TMEMBER** command, OTMA ignores the IMS Connect request. If the IMS Connect value is less than the value that is specified in OTMA, then the IMS Connect **MAXI** value is used. A specification of 0 is ignored because IMS Connect cannot disable the OTMA message flood control support. Disabling the capability can be done only in OTMA.

Value: This is an effective method to control flood conditions that can result in storage shortage situations.

2.5.4 IMS Version 10: Resume Tpipe port affinity

IMS Connect clients use the RESUME Tpipe protocol to retrieve the commit-then-send (CM0) output or synchronous callout requests from a Tpipe hold queue in IMS. The **PORTAFF** parameter in the **TCPIP** statement controls whether commit-then-send (CM0) output messages that are sent by IMS to an IMS Connect system have affinity to the port on which IMS Connect received the original input message as follows:

PORTAFF=Y	IMS Connect returns all CM0 output for this IMS Connect client through the same port on which it received the original input message.
PORTAFF=N	IMS Connect attempts to return the CM0 output to the first client it finds on any available port with an outstanding request from this client ID.

Note: If you are running a sysplex environment that has implemented redundancy, load balancing, super member support, and so on, **PORTAFF=N** is a reasonable choice. Using the same instance of a single client ID across multiple ports is not recommended.

Value: In situations where IMS Connect must maintain affinity to a specific client, this parameter is of use.

2.5.5 IMS Version 11: CM0 ACK timeout support

The **HWS** statement provides a new **CM0ATOQ** value that defines a global override timeout queue name for all the data stores (IMS systems) that are accessed by an IMS Connect instance. The IMS Connect **CM0ATOQ** value overrides the default value of **DFS\$STOQ** in OTMA.

Value: IMS Connect takes advantage of the OTMA CM0 timeout support and provides a new parameter **CM0ATOQ** that can be used to provide a timeout queue name.

2.5.6 IMS Version 11: Super member support at a data store level

The **DATASTORE** statement is enhanced to support a new parameter for super member support. The new **SMEMBER** parameter on individual **DATASTORE** statements overrides the default **HWS** value. By providing the super member support at a data store level, IMS Connect is now able to support multiple super member names.

Using a super member group, you can configure OTMA to send output messages to a remote IMS installation through up to eight local instances of IMS Connect.

The use of a super member group is transparent to both the IMS application that sends the messages and the remote IMS installation that receives the messages. Any participating hold-queue-capable client can retrieve the CM0 messages on the super member output queue by issuing its own RESUME Tpipe call, regardless of which client the CM0 output was originally destined for. The messages on the super member queue are retrieved on a first-in-first-out basis, without regard to which instance of the OTMA client originated the output or which instance of the OTMA client issued the RESUME Tpipe call.

Figure 2-9 illustrates the use of a super member group with an IMS-to-IMS TCP/IP connection:

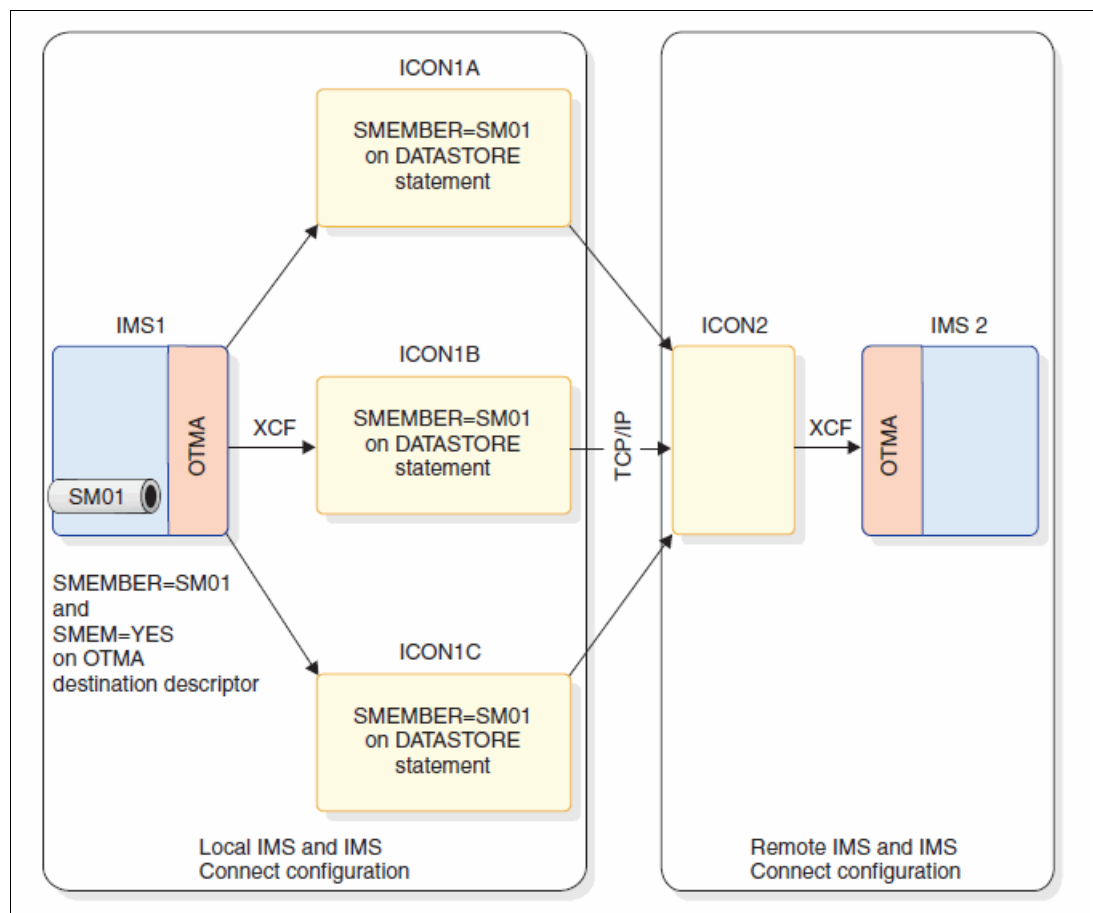


Figure 2-9 Sample super member

Value: Super member support provides parallel access paths for output messages. This improves scalability and performance.

2.5.7 IMS Version 13: Configuring the TCP/IP maximum backlog queue size

When IMS Connect starts, it establishes a TCP/IP listening socket on every port that is configured for its use. When these listening sockets are established, IMS Connect also sets the corresponding TCP/IP backlog or queue value to be equal to the **MAXSOC** value for IMS Connect. (The default for **MAXSOC**, if not specified, is 50). This backlog or queue value is the number of connection requests that can be queued in TCP/IP while waiting for IMS Connect to assign sockets to them, for example, during peak processing times. This means that requests up to this value can be queued in TCP/IP without their connections being rejected pending IMS Connect action.

If the IMS Connect value is higher than what is specified for **SOMAXCONN** in the TCP/IP profile, the smaller of the two values (**SOMAXCONN** or **MAXSOC**) is used by the TCP/IP stack.

IMS Connect Version 13 adds a parameter, **TCPIPQ**, to the configuration member for IMS Connect. The **SOMAXCONN** statement in **PROFILE.TCPIP** controls the TCP/IP queue depth for listening sockets at the LPAR level. You can use the **TCPIPQ** parameter to override the value of **SOMAXCONN** for an instance of IMS Connect.

TCPIPQ values can be 0 - 2147483647 with a default of 50.

- ▶ If a value of 0 to 49 is set, IMS Connect overrides the setting with default 50 value.
- ▶ If the **TCPIPQ** value is less than zero or larger than 2147383647, messages HWSX0909E and BPE003E are issued and IMS Connect abends with an U3401 code.
- ▶ If the **TCPIPQ** is greater than **SOMAXCONN**, the number that is used is the one defined for **SOMAXCONN**.

The new **TCPIPQ** parameter value should therefore be coordinated with the value in the **SOMAXCONN**. If **TCPIPQ** is not specified in the IMS Connect configuration definition, then the default that is used is the value that is specified in **MAXSOC**.

Value: You have more control over connection requests that can be queued and waiting for service.

2.6 IMS Connect user exits

Because each client message can be unique in design, format, processing, and security requirements, data manipulation of such messages into a minimum level of IMS Connect recognizable format is essential for the ability of IMS Connect to act as the go between for both the client and the data store.

To achieve this minimum level of recognizable format, IMS Connect provides support for user exits to receive and manipulate messages from both the originating client and the IMS server. These exits can be customized to perform various functions, such as reformatting, translation, and security verification. They can be regarded as plug-ins to IMS Connect, so the central or kernel of the IMS Connect logic is isolated from the message manipulation exits.

IBM delivers prewritten clients and their associated user exits for simplified development. However, customers can create and develop their own TCP/IP clients, with their associated user exits, according to their installation requirements and needs.

IBM delivers fifteen supported IMS Connect related user exits at the Version 13 level. Six of these exits are message exits. HWSIMSO0 and HWSIMSO1 were removed in IMS Version 11, so they are not included in Table 2-1.

Table 2-1 IMS Connect user exits that are available at the IMS Version 13 level

User exit name	Purpose
HWSSMPL0	This exit is based on HWSIMSO0. It performs translating input messages into the protocol or format that is required by IMS and OTMA, rerouting messages, checking security for input messages, and returning user-defined messages in response to certain user-defined criteria. Migrate to HWSSMPL1 because the new function is no longer added to HWSSMPL0.
HWSSMPL1	This exit is based on HWSIMSO0. It has a fullword length field proceeding the message.
HWSJAVA0	User exit in support of the IMS Connector for Java client.
HWSSOAP1	IBM delivered sample user exit in support of the IMS Enterprise Suite SOAP Gateway.

User exit name	Purpose
HWSCSLO0	IMSpIex support exit. Enables the IMS Connect capability to communicate with the IMS Operations Manager (OM). This is necessary to use the IMS Control Center.
HWSCSLO1	Same as HWSCSLO0 with some differences, such as it does not perform any translation output messages.
IMSLSECX	This is the z/OS TCP/IP IMS Listener security exit routine.
HWSYDRU0	The sample IMS Connect Destination Resolution exit routine (HWSYDRU0), which is a modified version of the OTMA Destination Resolution exit routine (DFSYDRU0).
HWSUINIT	Enables you to perform your own processing during IMS Connect initialization and termination. This exit receives control at initialization and termination time.
HWSTECL0	Enables you to customize IMS Connect to support event recording.
HWSPWCH0	IMS Connect password change exit routine.
HWSPIOX0 or user specified name	IMS Connect Port Message Edit exit routine for TCP/IP clients. Receives control between IMS Connect and the z/OS TCP/IP stack to modify the format of input and output messages. You specify a Port Message Edit exit routine as a 1- to 8-character name on the EDIT parameter of the PORT keyword in the TCP/IP configuration statement in the HWSCFGxx PROCLIB member.
HWSROUT0	IMS Connect DB Routing user exit routine.
HWSAUTH0	IMS Connect DB Security user exit routine.
HWSDPWR1	IBM WebSphere DataPower provided as object code only.

For more information about most of these exits, see *IMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity*: SG24-6974.

2.7 IMS Connect user exit enhancements

User exits are an important component of IMS Connect and enhancements to their effectiveness were introduced.

2.7.1 IMS Version 11: User message exit cleanup

HWSIMSO0 and HWSIMSO1 message user exits are no longer shipped with IMS Connect. HWSSMPL0 and HWSSMPL1 user exits provide enhanced functionality and are delivered as source code replacements, which also reduce the number of user message exits to maintain.

Value: Reduction in complexity over exit maintenance.

2.7.2 IMS Version 11: Modifying formats of incoming and outgoing messages

The IMS Connect Port Message Edit exit routine was introduced to provide you with an exit between TCP/IP and IMS Connect where you can modify the format of incoming and outgoing messages.

You specify a Port Message Edit exit routine as a 1- to 8-character name on the **EDIT** parameter of the **PORT** keyword in the **TCPIP** configuration statement in the **HWSCFGxx** PROCLIB member. The exit is specified on the IMS Connect port parameter so that you can use specialized exits on different ports as required by various IMS Connect clients.

Value: Increased control over the format of messages on different ports.

2.7.3 IMS Version 11: HWSEXPRL parameter list expansion

HWSEXPRL maps the parameter list that is passed to the user exit routine on each subroutine call. A copy of this macro is in **SDFSMLC**. To see the structure, assemble the macro.

IMS Version 11 expands the existing HWSEXPRL macro that is used by IMS Connect user message exit routines. You must reassemble and rebind the IMS Connect exit routines that use this macro. The exits that must be reassembled and rebound include the IMS Connect Sample Exits (HWSSMPL0 and HWSSMPL1) and, if it was modified, the IMS TM Resource Adapter exit routine (HWSJAVA0).

Value: The changes in HWSEXPRL result in the following items:

- ▶ Improved readability and serviceability of the parameter list.
- ▶ Reorganizes the current fields for better placement and eliminates discrepancies that are caused by lack of space.

2.7.4 IMS Version 12: Load modules packaging for exits

IBM supplies samples for some IMS Connect exits, including **HWSUINIT**, **HWSJAVA0**, **HWSSMPL0**, and **HWSSMPL1**. These exits are working examples, and many customers use the IBM supplied exits without modifications.

In IMS Version 12, the exits are repackaged as load modules and sample source. Users who want to modify the IBM supplied samples can still do so. Customers who assemble and bind the current samples without modification can remove a task from their migration and implementation plans.

If you must assemble the sample source for any reason, then the JCL that is used to for this task must include **IMS.SDFSMLC**, **SYS1.MACLIB**, and **SYS1.MODGEN**. Otherwise, the assembly fails.

Value: This feature reduces the effort in the migration and implementation of IMS systems.

2.8 IMS Connect high availability facilities

IMS Connect use mechanisms such as IP spraying, workload balancing, and sysplex distribution to allow a connection request to be routed to any of the available IMS Connect instances.

A Sysplex Distributor environment is an important component of high availability for IMS Connect. It is a component of the IBM z/OS Communication Server and implements a dynamic Virtual IP address (VIPA) as a single network-visible IP address for a set of hosts that belong to the same sysplex cluster. Any client anywhere in the IP network is able to see the sysplex cluster as one IP address regardless of the number of hosts that it includes. Here are its high availability features:

- ▶ Enhanced dynamic VIPA and automatic takeover
- ▶ Tracks connections, target stacks, and applications, such as IMS Connect
- ▶ Starts the services of the Workload Manager (WLM)
- ▶ Supports automatic VIPA takeover, allowing an active stack to assume the load of a failing stack

2.8.1 IMS Connect workload balancing and failover

IMS Connect can access multiple IMS systems (data stores) and message exits can reroute a message to a different target IMS. The data store table provides information as to which systems are active

Figure 2-10 presents these concepts in a graphical format.

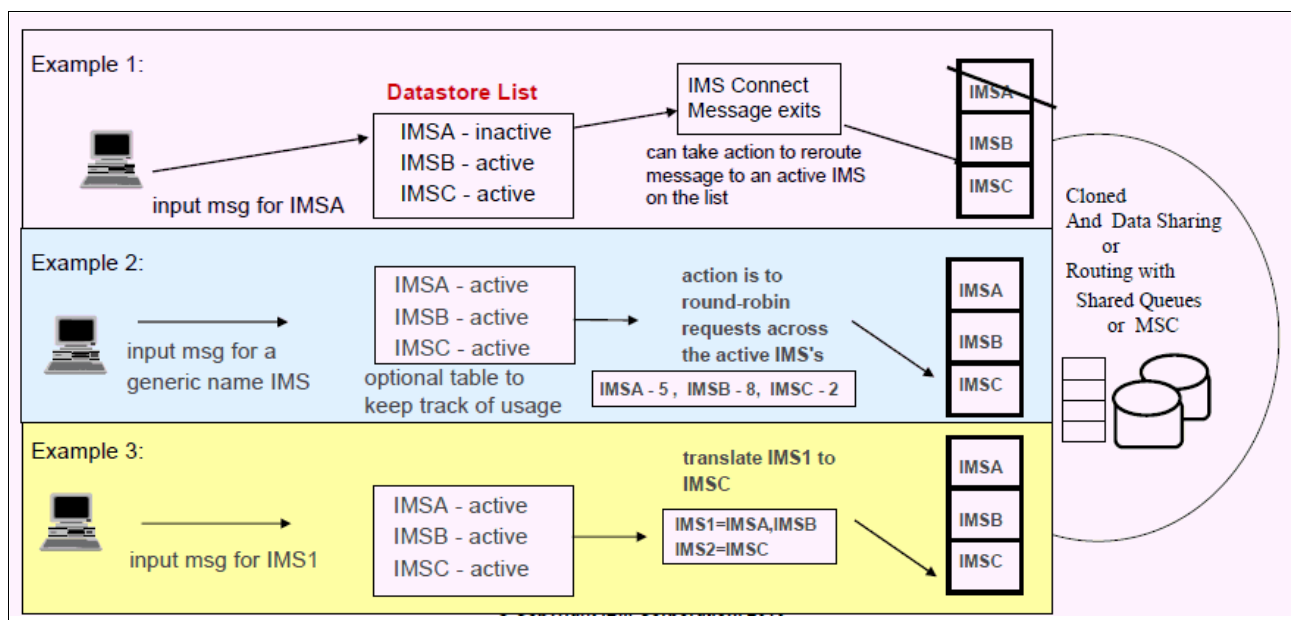


Figure 2-10 IMS Connect workload balancing and failover

There are two tables that are available to IMS Connect Message exits that are associated with this dynamic routing of messages:

- ▶ **INIT TABLE**
Points to the data store table and allows user data to be stored.
- ▶ **DATASTORE TABLE** (A data store is an IMS system)
Contains data store IDs, status (active or inactive), and optional user data.

A User Initialization Exit Routine (HWSUINIT) is available that is driven during IMS Connect initialization and termination. It loads user tables, obtains any needed storage, and adds user data (definitions of the alternative IMS systems) to the INIT or DATASTORE tables. IMS Connect provides the interface, but the base IMS message exits do not take advantage of the capability. These message exits can be enhanced to do so through plug-ins that are available from IMS Connect Extensions tools.

2.9 IMS Connect high availability enhancements

We examine two enhancements that were introduced:

- ▶ IMS Version 11: TCP/IP auto reconnect
- ▶ IMS Version 13: Reporting IMS Connect health to the Workload Manager

2.9.1 IMS Version 11: TCP/IP auto reconnect

If the Internet Protocol network experiences a disruption during IMS Connect processing, operators must issue the **OPENPORT** command when the network is once again available. With the TCP/IP auto reconnect support in IMS Version 11, the same situation no longer requires operator intervention because IMS Connect continues to listen on the socket that is associated with each active port and automatically attempts to re-establish a connection as appropriate.

Value: With TCP/IP Auto Reconnect, IMS Connect can continue to autonomically listen on the LISTEN SOCKET for the re-emergence of the network. When the network becomes active, IMS Connect re-establishes communication with TCP/IP without operator intervention.

2.9.2 IMS Version 13: Reporting IMS Connect health to the Workload Manager

The Workload Manager (WLM) provides facilities that allow servers to report their health to WLM. This health information can be used by the Sysplex Distributor and taken into account when connection requests are routed to a server.

IMS Connect Version 13 takes advantage of the facilities that are provided by WLM to inform WLM of its overall health. As the health of IMS Connect deteriorates or improves, IMS Connect notifies WLM so that WLM has the most recent health information available for use by the Sysplex Distributor. Sysplex Distributor uses the health of each server when the distribution method is set to SERVERWLM in the **VIPADISTRIBUTE** statement of the TCP/IP profile.

Figure 2-11 shows an overview of the WLM Health Report as it applies to IMS Connect.

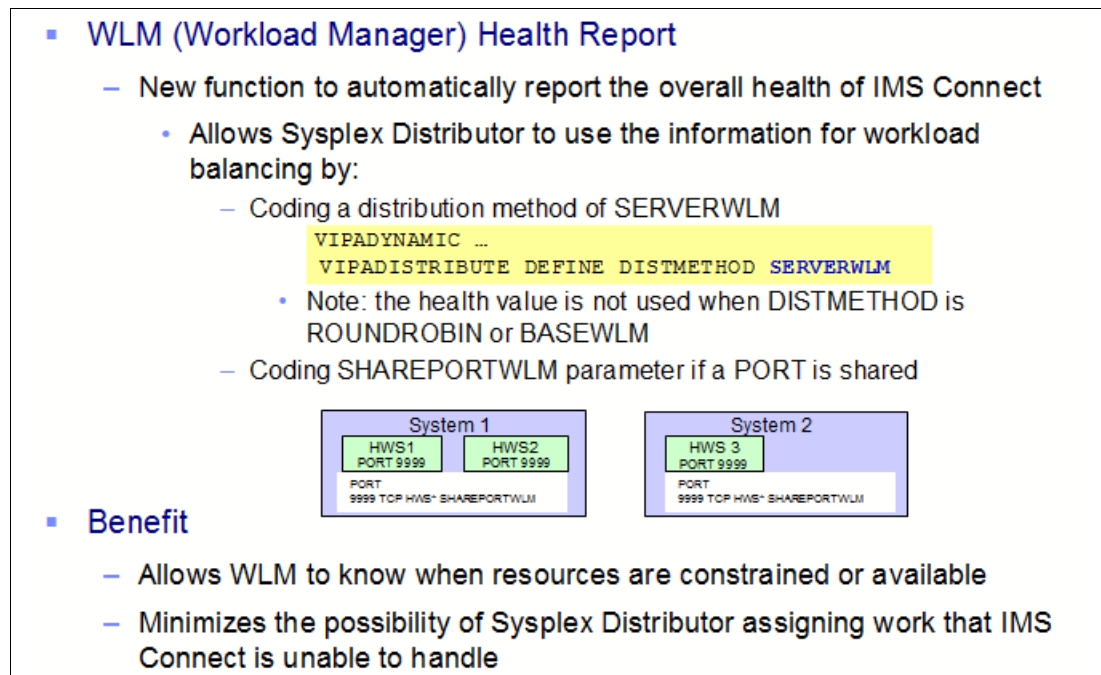


Figure 2-11 IMS Connect and the WLM Health Report

Incoming connections are routed to multiple stacks based on the defined distribution method:

- ▶ ROUNDROBIN requests an even distribution to all targets.
- ▶ BASEWLM uses WLM system weights.

IMS Connect automatically informs WLM of its overall health during initialization and any changes in its health afterward. IMS Connect is notified with ENF (event code) 71. The initial health is 100 and the minimum health can be 0. IMS Connect reports its health to WLM whenever the health increases or decreases by at least 5% from the last reported value.

For example, supposed the health is decreasing and the last reported health is 80%. Suppose that three actions take place that reduce the health of the IMS Connect by about 16%. Suppose two of the activities bring the health down by 10%. Because the reporting threshold is 5% and 10% is larger than or equal to 5%, IMS Connect reports its new health of 70% to WLM. The last activity takes place and the health is reduced by the last 6%, bringing the health down to 64%. Because 6% is larger than or equal to 5%, IMS Connect reports a change in health to WLM. The health is reported as 65% and not 64% because 65% is the last multiple of 5 that was encountered going down from 70% to 64%.

The opposite scenario where the health is increasing is as follows: If the last reported health is 50% and it increases to 64%, IMS Connect reports its health as 60%. The health is reported as 60% and not as 64% because 60% is the last multiple of 5 that was encountered going up from 50% to 64%.

The Sysplex Distributor and the TCP/IP stack can take advantage of the health value to determine which IMS Connect is the most viable for routing connection requests.

Value: IMS Connect can now report its overall health to the Workload Manager (WLM). This allows the Workload Distributor to autonomically balance workload in the most effective manner.

2.10 IMS Connect operations and command support

You can interact with IMS Connect by using three types of command interfaces:

- ▶ Commands that are entered as system replies against the IMS Connect started task.
- ▶ Commands that are entered as MODIFY instructions against the IMS Connect started task.
- ▶ Commands that are entered through the type-2 command interface from the IMS Single Point of Control facility (SPOC).

2.10.1 IMS Connect support for IMSplex and the IMS Control Center

IMS Connect can send and receive IMS Operations Manager (OM) commands and response string messages between an IMS Control Center client and the OM in an IMSplex by using the IMS Structured Call Interface (SCI). Figure 2-12 shows the components that are associated with the IMS Control Center and IMS Connect.

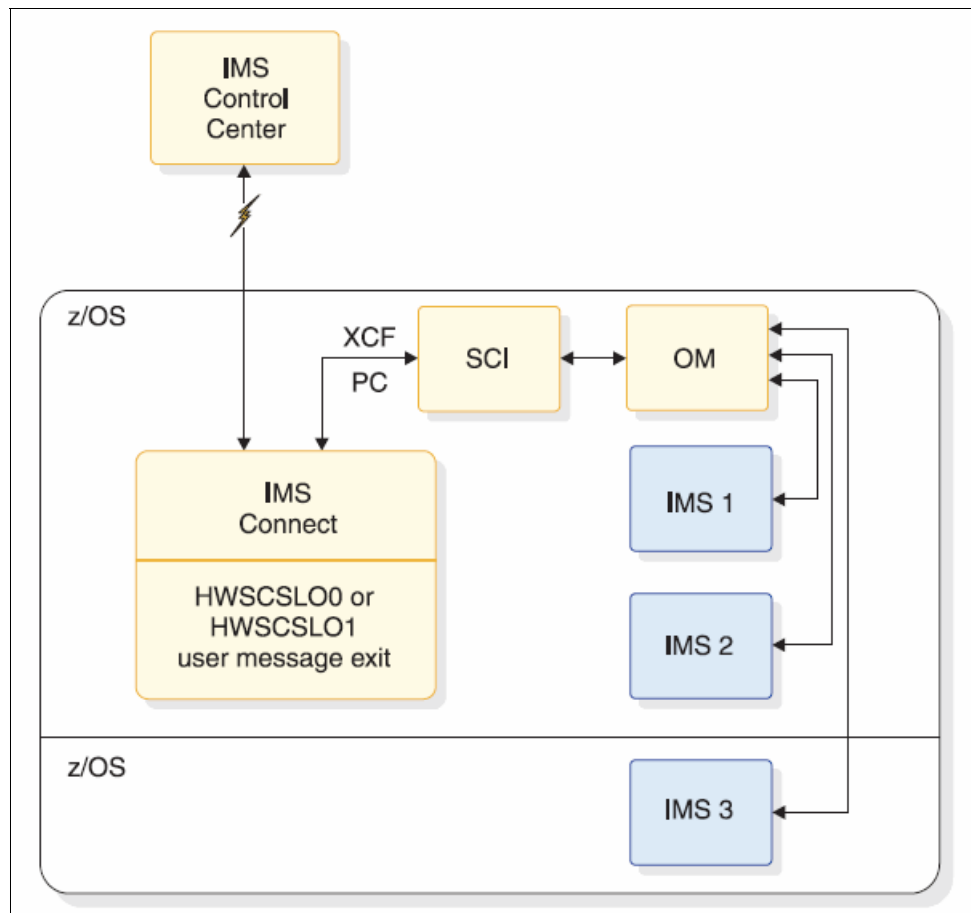


Figure 2-12 Overview of IMS Connect support for IMS Control Center

SCI allows IMSplex members to communicate with each other. The communication between IMSplex members can happen within a single z/OS image or among multiple z/OS images. Individual IMS components do not need to know where the other components are or what communication interface to use.

We now present the base set of commands. Commands that were introduced after IMS Version 9 are introduced in 2.11, “IMS Connect operations and command enhancements” on page 56.

2.10.2 IMS Connect REPLY commands

You can issue the following commands using a **REPLY** against the IMS Connect started task:

- ▶ **CLOSEHWS**
- ▶ **OPENDS** or **STARTDS**
- ▶ **OPENIP** or **STARTIP**
- ▶ **OPENPORT** or **STARTPT**
- ▶ **RECORDER**
- ▶ **SETRACF**
- ▶ **SETRRS**
- ▶ **STOPCLNT**
- ▶ **STOPDS**
- ▶ **STOIP**
- ▶ **STOPPORT**
- ▶ **VIEWDS**
- ▶ **VIEWHWS**
- ▶ **VIEWIP**
- ▶ **VIEWPORT**

All of these IMS Connect commands must be immediately preceded on the command line of the z/OS system console by the reply number of the outstanding IMS Connect reply message.

The following sections list basic IMS Connect commands with their parameters and an example of the output from each one. All the following examples are based on the IMS Connect configuration file in Example 2-10.

Note: There is a complete set of comments in this example that assist you in understanding the IMS Connect configuration parameters.

Example 2-10 Sample IMS Connect configuration member HWCF13B1

```
* -----HWS-----
* ID=
* PSWDMC=N MIXED-CASE PASSWORDS DISABLED
* CMOATOQ NAME OTMA CMO ACK TIMEOUT QUEUES(DEF DFS$TQ)
* CAN BE OVERRIDED BY DATASTORE
* RACF=N DISABLES SECURITY SUPPORT BY RACF
* RRS=Y RESOURCE RECOVERY SERVICES ENABLED
* SMEMBER SUPERMEMBER
* #UIDAGE= REFRESH INTERVAL FOR CACHED RACF USER IDS IN SECS
* #UIDCACHE=Y IMS CONNECT CACHES VERIFIED RACF USER IDS FOR REUSE
* XIBAREA= NUMBER OF FULLWORDS ALLOCATED FOR THE XIB USER AREA
HWS=(ID=HWSI13B1,PSWDMC=R,RACF=N,RRS=Y,
UIDCACHE=Y,UIDAGE=500,XIBAREA=50)
* -----TCP/IP-----
* EXIT NAMES OF ONE OR MORE IMS MESSAGE EXIT ROUTINES
* HWSJAVA0 IS AUTOMATICALLY LOADED
* IMS CONTROL CENTER REQUIRES HWCSL00 AND HWCSL01
* HOSTNAME TCP/IP JOBNAME
```

```

*   PORTID      TCP/IP PORT ENABLES LOCAL OPTION CONNECTIONS
*   -- OR --
*   PORT        TCP/IP PORT
*   SSLPORT     TCP/IP PORT FOR SSL SUPPORT
*   SSLENVAR    NAME OF THE SSL INITIALIZATION FILE
TCPIP=(EXIT=(HWSSMPL1,HWSSOAP1),HOSTNAME=TCPIP,PORT(ID=7200),
PORT(ID=7201),PORT(ID=7202))
* -----DATASTORE -----
* DATASTORE ADDRESSES IMS CONTROL REGION VIA OTMA
*   GROUP      XCF(OTMA) FROM GRNAME IN DFSPBXXX
*   ID         DATA STORE NAME (IN IRM REQUEST)
*   MEMBER     XCF MEMBER NAME THAT IDENTIFIES IMS CONNECT
*   IF SAME GROUP, NAME MUST BE UNIQUE WITHIN DATASTORE
*   TMEMBER    THE XCF MEMBER NAME FOR IMS
*   OAAV=      (ACEE) AGING VALUE, IN SECONDS (DEFAULT 999999)
*   MAXI=      INPUT MESSAGE FLOOD CONTROL VALUE
*   CMOATOQ=   OTMA CMO ACK TIMEOUT QUEUE (DEF DFS$TOQ)
*   APPL=      TCP/IP APPL NAME DEFINED TO RACF IN PTKTDATA
*   DRU=       OWN OTMA DESTINATION RESOLUTION USER EXIT NAME
*   RRNAME=    NAME OF AN ALTERNATE DESTINATION SPECIFIED IN A CLIENT
*   ACKTO=     TIMEOUT INTERVAL FOR OUTPUT ACKS.CMO,CM1..IMS TO IMS
*   SMEMBER    SUPERMEMBER
DATASTORE=(GROUP=I13XOTMA,ID=I13B,MEMBER=HWBI13B2,TEMBER=I13BOTMA)
* -----ADAPTER-----
*   XML=Y      XML ADAPTER SUPPORT SHOULD BE ENABLED (SOAP GATEWAY)
ADAPTER=(XML=Y)
* -----IMSPLEX-----
* -- REGISTERS IMS CONNECT AS A MEMBER OF AN IMPLEX
* -- ENABLES BOTH IMS TYPE-2 COMMAND SUPPORT FOR IMS CONNECT AND
* -- COMMUNICATIION BETWEEN IMS CONNECT AND OTHER MEMBERS OF THE IMSPLEX
*   MEMBER     NAME THAT IDENTIFIES IMS CONNECT IN THE IMSPLEX
*   TMEMBER    IMSPLEX THAT IMSCON IS JOINING, IMSPLEX(NAME= )
IMSPLEX=(MEMBER=HWSI13B1,TEMBER=PLX13)
* -----ODACCESS-----
*   DRDAPORT   PORT NUMBER, TCP/IP KEEP ALIVE VALUE,
*   (ID        PORTNR
*   KEEPAV     TCP/IP LAYER SENDS A PACKET ON IDLE CONS
*   PORTTMOT   AMOUNT TIME IMS CONNECT WAITS FOR NEXT INPUT)
*   ODBMAUTOCONN SPECIFIES WHETHER IMS CONNECT AUTOMATICALLY
*   NEW AND EXISTING INSTANCES OF ODBM WITHIN AN IMSPLEX.
ODACCESS=(DRDAPORT=(ID=6200,KEEPAV=0,PORTTMOT=18000),
DRDAPORT=(ID=6201,KEEPAV=0,PORTTMOT=18000),ODBMAUTOCONN=Y)
* #####-----MSC-----
*   GENIMSID=  MSC TCP/IP GENERIC IMS ID (OPTIONAL)
*   ALSO SPECIFIES ON THE GENIMSID IN DFSDCXXX
*   IMSPLEX (MEMBER, TMEMBER)
*   THIS NAME MUST MATCH THE NAME SPECIFIED ON THE LCLICON
*   PARM OF THE MSPLINK MACRO DEFINITION OF THE LOCAL IMS SYSTEM
*   LCLIMS=    IMS ID OF THE LOCAL IMS SYSTEM AS REGISTERED WITH SCI
*   LCLPKLID=  LOCAL NAME OF THE MSC PHYSICAL LINK
*   RMTIMS=    IMS ID OF THE REMOTE IMS SYST*          ==>QUERY IMSPLEX    ?? C
*   RMTIMSCON= REMOTE IMS CONNECT CONNECTION TO USE FOR MSC MESSAGES
*   MUST MATCH ID PARAMETER OF ONE OF THE
*   RMTIMSCON STATEMENTS SPECIFIED IN THE LOCAL IMS CONNECT
*   RMTPLKID=  NAME MSC PHYSICAL LINK IN REMOTE LCLPLKID

```

```

*
MSC=(LCLPLKID=MSC2B2A,RMTPLKID=MSC2A2B,LCLIMS=(I13B),RMTIMS=ACIM,
IMSPLEX=(MEMBER=HWSI13B1,TMEMBER=PLX13),RMTIMSCON=HWSI13A1)
*
*#####-----RMTIMSCON-----
* -CONNECTION DEFINED BY RMTIMSCON CAN BE USED FOR EITHER OTMA
* - OR MSC MESSAGES, BUT NOT BOTH
*      :OTMA REQUIRES IMSCON STATEMENT BY SENDING IMSCONN
*      :MSC REQUIRES CORRESPONDING RMTIMSCON IN OTHER HWSCFGXX
*      APPL=      APPLICATION NAME TO USE IN A RACF PASSTICKET
*      ID         IDENTIFIES THIS DEFINITION OF A CONNECTION
*      AUTOCONN=  CONNECTS TO REMOTE IMSCON DURING STARTUP (DEF=N)
*      IDLETO=    OPEN SOCKET CONNECTIONS IDLE TIME
*      *EITHER THE IPADDR OR THE HOSTNAME PARAMETER
*      IPADDR=    IP_ADDRESS OF OTHER IMS CONNECT
*      HOSTNAME   THE HOST NAME OF THE REMOTE IMS CONNECT
*      PERSISTENT=DEFINES SOCKETS USED FOR CONNECTION AS PERSIST
*      PORT=      PORTNR OF LISTENING PORT
*      RESVSOC=   NUMBER OF SEND SOCKETS THAT IMS CONNECT RESERVES
*      TO I13A VIA TCP/IP
*      -USED BY MSC
*
RMTIMSCON=(ID=HWSI13A1,HOSTNAME=WTSC63.ITSO.IBM.COM,
PORT=7102,AUTOCONN=Y,RESVSOC=10)
*
* -USED BY OTMA (these definitions have been commented out)
*
*RMTIMSCON=(ID=HWSI13A1,HOSTNAME=WTSC63.ITSO.IBM.COM,
*  PORT=7401,AUTOCONN=N,PERSISTENT=Y,IDLETO=60000,RESVSOC=10)
*RMTIMSCON=(ID=HWSI13C1,HOSTNAME=WTSC63.ITSO.IBM.COM,
*  PORT=7401,AUTOCONN=N,PERSISTENT=Y,IDLETO=60000,RESVSOC=10)
*RMTIMSCON=(ID=HWSI13D1,HOSTNAME=WTSC64.ITSO.IBM.COM,
*  PORT=7401,AUTOCONN=N,PERSISTENT=Y,IDLETO=60000,RESVSOC=10)
*-----RUNOPTS-----
*      RUNOPTS=LE_RUNTIME_OPTIONS
*-----***** END   HWC13B1   DEFINITION-----

```

CLOSEHWS

The **CLOSEHWS** command terminates IMS Connect. Here are the parameters for this command:

► QUIESCE

Specifies that termination ends all client and data store connections in a controlled manner. If no parameter is specified for **CLOSEHWS**, this parameter is used by default.

All work that is in progress, or that is queued for processing, is completed before IMS Connect is terminated.

► FORCE

Specifies that termination ends all client and data store connections immediately, which forces any IMS applications that are running for the connected clients to abnormally terminate.

Example 2-11 shows the output of this command.

Example 2-11 An example of the CLOSEHWS command

```
R 176,CLOSEHWS
IEE600I REPLY TO 176 IS;CLOSEHWS
HWST3505I COMMUNICATIONS WITH REMOTE IMS CONNECT HWSI13A1 STOPPED;
M=TSCH
HWSP1415I TCP/IP SOCKET FUNCTION CALL FAILED; F=ACCEPT4 , R=-1,
E=1152, M=SDCO, ID=7200
HWSP1415I TCP/IP SOCKET FUNCTION CALL FAILED; F=ACCEPT4 , R=-1,
E=1152, M=SDCO, ID=7201
HWSS0770I LISTENING ON PORT=7201      TERMINATED; M=SSCH
HWSP1415I TCP/IP SOCKET FUNCTION CALL FAILED; F=ACCEPT4 , R=-1,
E=1152, M=SDCO, ID=6200
HWSP1415I TCP/IP SOCKET FUNCTION CALL FAILED; F=ACCEPT4 , R=-1,
E=1152, M=SDCO, ID=7202
HWSS0770I LISTENING ON PORT=7202      TERMINATED; M=SSCH
HWSP1415I TCP/IP SOCKET FUNCTION CALL FAILED; F=ACCEPT4 , R=-1,
E=1152, M=SDCO, ID=6201
HWSS0770I LISTENING ON PORT=6201      TERMINATED; M=SSCH
HWSS0770I LISTENING ON PORT=7200      TERMINATED; M=SSCH
HWSS0770I LISTENING ON PORT=6200      TERMINATED; M=SSCH
ATR169I RRS HAS UNSET EXITS FOR RESOURCE MANAGER 742
HWS.HWSI13B1V131.SVL.SANJOSE.IBM REASON: UNREGISTERED
HWSS0781I TCPIP COMMUNICATION FUNCTION CLOSED; M=SOCC
HWSD0260I DS=I13B      TRANSMIT THREAD TERMINATED; M=DXMT
HWSD0260I DS=I13B      RECEIVE  THREAD TERMINATED; M=DREC
HWSM0560I IMSPLEX=PLX13  TRANSMIT THREAD TERMINATED; M=OXMT
HWSM0560I IMSPLEX=PLX13  RECEIVE  THREAD TERMINATED; M=OREC
HWSN1960I ODBM=IMS13BOD TRANSMIT THREAD TERMINATED; M=NXMT
HWSD0282I COMMUNICATION WITH DS=I13B      CLOSED; M=DSCL
HWSN1960I ODBM=IMS13BOD RECEIVE  THREAD TERMINATED; M=NREC
HWSM0582I COMMUNICATION WITH IMSPLEX=PLX13  CLOSED; M=DSCL
HWSN1985I COMMUNICATION WITH ODBM=IMS13BOD CLOSED; M=DSCL
HWSF3360I THE TRANSMIT THREAD TERMINATED FOR MSC PHYSICAL LINK
MSC2B2A ; M=IXMT
HWSF3360I THE RECEIVE  THREAD TERMINATED FOR MSC PHYSICAL LINK
MSC2B2A ; M=IREC
HWSD0282I COMMUNICATION WITH DS=MSC2B2A  CLOSED; M=DSCL
HWSD0282I COMMUNICATION WITH DS=MSC2B2A  CLOSED; M=DSCL
HWSD0280I DATASTORE COMMUNICATION FUNCTION CLOSED; M=DOC3
HWSC0020I IMS CONNECT IN TERMINATION
BPE0007I HWS  BEGINNING PHASE 1 OF SHUTDOWN
IEE400I THESE MESSAGES CANCELLED - 177.
BPE0008I HWS  BEGINNING PHASE 2 OF SHUTDOWN
BPE0046I EXTERNAL TRACE DATA SET IMS13Q.IMS13B.HWS.RCTR.G0007V00
CLOSED ON VOL=SBOXI6
BPE0009I HWS  SHUTDOWN COMPLETE
$HASP395 IM13BHW1 ENDED
```

Starting an IMS Connect job

To start an IMS Connect job, run `/S IMS Connect jobname`. In our example, we ran `/S IM13BHW1`. Example 2-12 presents these messages from our example command.

Example 2-12 Example messages that are associated with an IMS Connect initialization

```
S IM13BHW1
$HASP100 IM13BHW1 ON STCINRDR
IEF695I START IM13BHW1 WITH JOBNAME IM13BHW1 IS ASSIGNED TO USER STC
, GROUP SYS1
$HASP373 IM13BHW1 STARTED
IEF403I IM13BHW1 - STARTED - TIME=23.49.18 - ASID=009D - SC64
BPE0046I EXTERNAL TRACE DATA SET IMS13Q.IMS13B.HWS.RCTR.G0008V00
OPENED ON VOL=SB0XI6
HWSX0920W VALUE OF PARAMETER AUTOCONN IN STATEMENT RMTIMSCON
=HWSI13A 779
CHANGED TO N FROM Y BECAUSE PERSISTENT=N WAS SPECIFIED ; M=XCFG
HWSX0920W VALUE OF PARAMETER PERSISTENT IN STATEMENT RMTIMSCON
=HWSI13A 779
CHANGED TO N FROM Y BECAUSE PERSISTENT=N WAS SPECIFIED ; M=XCFG
HWSX0920W VALUE OF PARAMETER PERSISTENT IN STATEMENT RMTIMSCON
=HWSI13A 780
CHANGED TO Y FROM N BECAUSE THE RMTIMSCON IS USED BY MSC; M=XCFG
HWS01210W IXCQUERY FAILED FOR GROUP=I13XOTMA, MEMBER=I13B0TMA
; R=8, S=4, M=DDX0
HWSM0590I CONNECTED TO IMSPLEX=PLX13 ; M=OSC1

HWS0292I CONNECTION TO DATASTORE=I13B , FAILED; M=DSC1
HWSC0010I WELCOME TO IMS CONNECT!
HWSF3300I COMMUNICATIONS ON MSC PHYSICAL LINK MSC2B2A STARTED; M=ISC1
HWSI1650I COMMAND REGISTRATION SUCCESSFUL FOR IMSPLEX=PLX13 ; M=OREG
HWSR0653I PROTECTED CONVERSATION PROCESSING WITH RRS/MVS ENABLED
M=RRSI
HWSS0780I TCPIP COMMUNICATION ON HOSTNAME=TCPIP OPENED; M=SDOT
HWSS0790I LISTENING ON PORT=7200 STARTED; M=SDOT
HWSS0790I LISTENING ON PORT=7201 STARTED; M=SDOT
HWSS0790I LISTENING ON PORT=6200 STARTED; M=SDOT
HWSS0790I LISTENING ON PORT=7202 STARTED; M=SDOT
HWSS0790I LISTENING ON PORT=6201 STARTED; M=SDOT
HWSN1900I IMS CONNECT IS CONNECTED TO ODBM=IMS13BOD; M=NSC1
HWST3500I COMMUNICATIONS WITH REMOTE IMS CONNECT HWSI13A1 STARTED;
M=TSCH
```

OPENDS or STARTDS

The **OPENDS** and **STARTDS** commands are equivalent and can be used to start communication between IMS Connect and a data store. The parameter for this command is **datastore_id**, which specifies the name of the data store, as defined in the `HWSCFGxx` configuration member. Example 2-13 shows the output of this command.

Example 2-13 An example of the OPENDS command

```
R 122,OPENDS I13B
IEE600I REPLY TO 122 IS;OPENDS I13B
HWS0230I DS=I13B ALREADY ACTIVE, R=0, S=ACTIVDST, M=DOCM
```

OPENIP or STARTIP

The **OPENIP** or **STARTIP** commands can be used to start or re-establish the communication between IMS Connect and the IMSplex that contains the Operations Manager (OM) instance that is connected to SCI and is used to issue commands.

The parameter for this command is **imsplex_id**, which identifies the IMSplex. This name must be defined to IMS Connect through the configuration member HWSCFGxx and must match the **TMEMBER** that is defined in the IMSplex configuration statement.

Example 2-14 shows the output of the **OPENIP** command.

Example 2-14 Output of the OPENIP command

```
R 175,OPENIP PLX13
IEE600I REPLY TO 175 IS;OPENIP PLX13
HWSM0590I CONNECTED TO IMSPLEX=PLX13 ; M=OSC1
HWSI1650I COMMAND REGISTRATION SUCCESSFUL FOR IMSPLEX=PLX13 ; M=OREG
```

OPENPORT or STARTP

The **OPENPORT** and **STARTP** commands can be used to reestablish IMS Connect communication with TCP/IP to enable listening on TCP/IP ports. The parameter for this command is **portid**, which identifies the number of the port to be opened. For the local option port, specify a port ID value of LOCAL.

Example 2-15 shows the output of this command after a **STOPPORT 7200** command is issued, as described in “STOPPORT” on page 51.

Example 2-15 Output of the OPENPORT command

```
R 173,OPENPORT 7200
IEE600I REPLY TO 173 IS;OPENPORT 7200
HWSS0790I LISTENING ON PORT=7200 STARTED; M=SDOT
```

RECORDER

The **RECORDER** command can either open or close the recorder trace file. This file is a reusable file and requires an IDCAMS VSAM REPRO utility to dump the contents for analysis. It is used primarily to diagnose problems with the IMS Connect user exits.

Allocate the HWSRCORD data set as a normal physical sequential data set with record format FB, record length 1,440, and block size 14,400. The data set can use secondary extents. IMS Connect stores the trace information into this single data set. There are no spare data sets or any wrap-around processing, so when this data set fills up, the trace is disabled. Starting the trace overwrites trace data that is recorded in an earlier trace.

From an IMS perspective, you can manage this trace data set in a similar manner as with the IMS Monitor Trace (IMSMON) data set.

The data set must be identified by an **HWSRCORD DD** statement in the IMS Connect startup JCL. Here are the parameters for this command:

- ▶ **OPEN**
Opens the recorder trace.
- ▶ **CLOSE**
Closes the recorder trace.

Because an **HWSRCORD DD** statement was not included in the test environment that we used, we received message IEC130I HWSRCORD DD STATEMENT MISSING. But, Example 2-16 shows the successful output of this command when entered from another system that does have the **HWSRCORD DD** statement included.

Example 2-16 Output of the RECORDER command

```
R 83, RECORDER OPEN
*084 HWSC0000I *IMS CONNECT READY* HWSI13B1
HWSR0880I RECORDER OPENED; M=RCDR
R 84, RECORDER CLOSE
*085 HWSC0000I *IMS CONNECT READY* HWSI13B1
HWSR0890I RECORDER CLOSED; M=RCDR
```

SETRACF

The **SETRACF** command can turn on or off the RACF flag and any subsequent RACF user ID checking. The **RACF=** parameter in the HWS startup parameter sets this at IMS Connect start. Here are the parameters for this command:

- ▶ **ON**
Turns on RACF security.
- ▶ **OFF**
Turns off RACF security.

Example 2-17 shows the output of this command.

Example 2-17 An example of the SETRACF command

```
R 149,SETRACF ON
IEE600I REPLY TO 149 IS;SETRACF ON
R 160,SETRACF OFF
IEE600I REPLY TO 160 IS;SETRACF OFF
```

After the **SETRACF ON** command is run, a **VIEWHWS** indicates that **RACF=Y** has been set and after the **SETRACF OFF** completes, a **VIEWHWS** displays **RACF=N**.

SETRRS

The **SETRRS** command enables or disables communication between IMS Connect and RRS.

Here are the parameters for this command:

- ▶ **ON**
Turns on communication with RRS.
- ▶ **OFF**
Turns off communication with RRS.

Example 2-18 shows the output of the **SETRRS** command. The **VIEWHWS** output shows **RRS=N** and then **Y** after the **SETRRS OFF** and **SETRRS ON** commands complete, as expected.

Example 2-18 An example of the SETRRS command

```
R 162,SETRRS OFF
IEE600I REPLY TO 162 IS;SETRRS OFF
ATR169I RRS HAS UNSET EXITS FOR RESOURCE MANAGER 457
```

```
HWS.HWSI13B1V131.SVL.SANJOSE.IBM REASON: UNREGISTERED
```

```
R 163,SETRRS ON  
IEE600I REPLY TO 163 IS;SETRRS ON  
HWSC0000I PROTECTED CONVERSATION PROCESSING WITH RRS/MVS ENABLED  
M=RRSI
```

STOPCLNT

The **STOPCLNT** command terminates communication with a client by using a specific TCP/IP port. Here are the parameters for this command:

► **portid**

Identifies the number of the port whose client communication will be stopped.

► **clientid**

Identifies the name of the client, which is obtained from the **VIEWHWS** command.

Example 2-19 shows the output of this command (HWS8PBTM is the client name, which in this case is automatically generated by IMS Connector for Java). The M=SCCM output indicates that IMS Connect module HWSSCCM0 issued this response.

Example 2-19 An example of the STOPCLNT command

```
R 88, STOPCLNT 7003 HWS8PBTM  
*089 HWSC0000I *IMS CONNECT READY* HWSI13B1  
HWSS0761I TCP/IP COMMUNICATION WITH CLIENT=7003 _HWS8PBTM STOPPED;  
M=SCCM
```

STOPDS

The **STOPDS** command immediately terminates communication between IMS Connect and the data store. The parameter for this command is **datastore_id**, which specifies the name of the data store, as defined in the HWSCFGxx configuration member.

Example 2-20 shows the output of this command.

Example 2-20 An example of the STOPDS command

```
R 168,STOPDS I13B  
IEE600I REPLY TO 168 IS;STOPDS I13B  
HWSD0260I DS=I13B TRANSMIT THREAD TERMINATED; M=DXMT  
HWSD0260I DS=I13B RECEIVE THREAD TERMINATED; M=DREC  
HWSD0284I COMMUNICATION WITH DS=I13B STOPPED; M=DSCM
```

STOPIP

The **STOPIP** command stops communication between IMS Connect and the IMSplex that contains the OM that is connected to SCI that is used to send commands that are entered through the IMS SPOC. The parameter for this command is **imsplex_id**, which specifies the name of the IMSplex. This name must be defined to IMS Connect through the configuration member HWSCFGxx. In our example environment, that member is HWCF13B, which contains parameter **TMEMBER=PLX13**, as shown in Example 2-10 on page 43.

Example 2-21 shows the output of the **STOPIP** command.

Example 2-21 An example of the STOPIP command

```
R 170,STOPIP PLX13
IEE600I REPLY TO 170 IS;STOPIP PLX13
HWSN1960I ODBM=IMS13BOD TRANSMIT THREAD TERMINATED; M=NXMT
HWSN1960I ODBM=IMS13BOD RECEIVE  THREAD TERMINATED; M=NREC
```

STOPPORT

The **STOPPORT** command immediately terminates listening on a TCP/IP port. The parameter for this command is **portid**, which identifies the number of the port to be stopped.

Example 2-22 shows the output of the **STOPPORT** command.

Example 2-22 An example of the STOPPORT command

```
R 172,STOPPORT 7200
IEE600I REPLY TO 172 IS;STOPPORT 7200
HWSP1415E TCP/IP SOCKET FUNCTION CALL FAILED; F=ACCEPT4 , R=-1,
E=1152, M=SDCO, ID=7200
HWSS0770I LISTENING ON PORT=7200      TERMINATED; M=SSCH
```

VIEWDS

The **VIEWDS** command displays the status of the specified IMS system (data store). The parameter for this command is **datastore_id**, which specifies the name of the data store, as defined in the HWSCFGxx configuration member. In our example, the IMS **datastore_id** is I13B.

Example 2-23 shows an example of the output of this command.

Example 2-23 An example of the VIEWDS command

```
R 127,VIEWDS I13B
IEE600I REPLY TO 127 IS;VIEWDS I13B
      DATASTORE=I13B      STATUS=DISCONNECT
      GROUP=I13XOTMA MEMBER=HWBI13B2
      TARGET MEMBER=I13BOTMA      STATE=N/A
      DEFAULT REROUTE NAME=HWS$DEF
      RACF APPL NAME=
      OTMA ACEE AGING VALUE=999999
      OTMA ACK TIMEOUT VALUE=120
      OTMA MAX INPUT MESSAGE=5000
      SUPER MEMBER NAME=      CMO ACK TOQ=
```

VIEWHWS

The **VIEWHWS** data stores command displays the status of the OTMA connection.

Example 2-24 shows an example of the output of this command.

Example 2-24 An example of the VIEWHWS command

```
R 125,VIEWHWS
IEE600I REPLY TO 125 IS;VIEWHWS
      HWS ID=HWSI13B1 RACF=N  PSWDMC=R
      UIDCACHE=Y  UIDAGE=500
      MAXSOC=50  TIMEOUT=0  TCPIPQ=50
```

NUMSOC=5 WARNSOC=80% WARNINC=5%
 RRS=Y STATUS=ACTIVE
 VERSION=V13 IP-ADDRESS=009.012.006.009
 SUPER MEMBER NAME= CMO ACK TOQ=
 ADAPTER=Y MAXCVRT=100 NUMCVRT=0
 MAXLSSSZ=32767
 ODBM AUTO CONNECTION=Y
 ODBM TIMEOUT=18000
 ODBM IMSPLEX MEMBER= TARGET MEMBER=
 DATASTORE=I13B STATUS=DISCONNECT
 GROUP=I13XOTMA MEMBER=HWBI13B2
 TARGET MEMBER=I13BOTMA STATE=N/A
 DEFAULT REROUTE NAME=HWS\$DEF
 RACF APPL NAME=
 OTMA ACEE AGING VALUE=999999
 OTMA ACK TIMEOUT VALUE=120
 OTMA MAX INPUT MESSAGE=5000
 SUPER MEMBER NAME= CMO ACK TOQ=
 IMSPLEX=PLX13 STATUS=ACTIVE
 MEMBER=HWSI13B1 TARGET=PLX13
 MSC=MSC2B2A STATUS=ACTIVE
 RMTPLKID=MSC2A2B
 LCLIMSID=I13B RMTIMSID=ACIM
 GENIMSID= AFFINITY=
 IMSPLEX=PLX13
 MEMBER=HWSI13B1 TARGET MEMBER=PLX13
 RMTIMSCON=HWSI13A1
 IP-ADDRESS=009.012.006.070 PORT=7102
 HOSTNAME=WTSC63.ITS0.IBM.COM

NO ACTIVE LINK
 ODBM=IMS13BOD STATUS=REGISTERED ODBMRRS=Y
 ALIAS=I13B STATUS=ACTIVE
 ALIAS=I13X STATUS=ACTIVE
 NO ACTIVE ISC
 PORT=7200 STATUS=ACTIVE KEEPAV=0 NUMSOC=1 EDIT=
 TIMEOUT=0
 NO ACTIVE CLIENTS
 PORT=7201 STATUS=ACTIVE KEEPAV=0 NUMSOC=1 EDIT=
 TIMEOUT=0
 NO ACTIVE CLIENTS
 PORT=7202 STATUS=ACTIVE KEEPAV=0 NUMSOC=1 EDIT=
 TIMEOUT=0
 NO ACTIVE CLIENTS
 PORT=6200D STATUS=ACTIVE KEEPAV=0 NUMSOC=1
 EDIT= TIMEOUT=18000
 NO ACTIVE CLIENTS
 PORT=6201D STATUS=ACTIVE KEEPAV=0 NUMSOC=1
 EDIT= TIMEOUT=18000
 NO ACTIVE CLIENTS
 RMTIMSCON=HWSI13A1 STATUS=NOT ACTIVE
 IP-ADDRESS=009.012.006.070 PORT=7102
 HOSTNAME=WTSC63.ITS0.IBM.COM

 AUTOCONN=N PERSISTENT=Y

```
IDLETO=0
RESVSOC=10  NUMSOC=0
NO ACTIVE CLIENTS
NO ACTIVE RMTICIS
```

VIEWIP

The **VIEWIP** command displays the current activity for the IMSplex. This command parameter is **imsplex_id**, which species the name of the IMSplex for which information is displayed. This is an optional parameter, and if it is specified, it must match the ID parameter of the IMSplex configuration statement in the HWSCFGxx member.

Example 2-25 shows the output of the **VIEWIP** command.

Example 2-25 An example of the VIEWIP command

```
R 128,VIEWIP PLX13
IEE600I  REPLY TO 128 IS;VIEWIP PLX13
        IMSPLEX=PLX13  STATUS=ACTIVE
        MEMBER=HWSI13B1 TARGET=PLX13
```

Where:

- ▶ PLX13 is the name of the IMSplex that is defined in the ID parameter of the IMSplex configuration statement in the HWSCFGxx member.
- ▶ The status of the IMSplex is ACTIVE. It can also be NOT ACTIVE or DISCONNECT.
- ▶ HWSI13B1 is the name of the member as defined in the MEMBER parameter of the IMSplex configuration statement in HWSCFGxx.
- ▶ PLX13 is the target member name, that is, the member of the IMSplex to which IMS Connect has connected as HWSI13B1, as shown in Example 2-10 on page 43.

VIEWPORT

The **VIEWPORT** command displays the status of the communication between IMS Connect and the specified port. The parameter for this command is **portid**, which identifies the number of the port to be displayed.

Example 2-26 shows the output of this command.

Example 2-26 An example of the VIEWPORT command

```
R 129,VIEWPORT 7200
IEE600I  REPLY TO 129 IS;VIEWPORT 7200
        PORT=7200  STATUS=ACTIVE  KEEPAV=0  NUMSOC=1  EDIT=
        TIMEOUT=0
        NO ACTIVE CLIENTS
```

This display shows the following characteristics:

- ▶ The port number is 7200, as defined to IMS Connect.
- ▶ There are no active clients, but the connection is active.

2.10.3 IMS Connect MODIFY commands

You can issue commands to IMS Connect by using the operating system **MODIFY** interface. Here is the syntax of the **MODIFY** command:

```
/F jobname,command
```

Jobname is the name of the IMS Connect started task, and **command** is the string of characters you want IMS Connect to run. For example, to issue the equivalent of a **VIEWHWS** reply command, you issue the following command:

```
/F IM13BHW1,QUERY MEMBER TYPE(IMSCON)
```

IMS13BHW1 is the job name of the IMS Connect.

This interface supports the use of wildcard characters:

- * Matches zero or more characters.
- % Matches exactly one character.

The wildcard characters are the same ones that you use in TSO and PDF when you specify a member or data set names.

The commands that you can issue using the **MODIFY** interface are functionally equivalent to the ones you can issue using **REPLY**. The commands responses are also the same.

Table 2-2 shows the equivalence between the **REPLY** syntax and the **MODIFY** commands.

Table 2-2 Equivalence between REPLY and MODIFY commands

REPLY command	MODIFY command
CLOSEHWS	SHUTDOWN MEMBER OPTION({QUIESCE FORCE})
OPENDS/STARTDS	UPDATE DATASTORE (name1,name2,...) START(COMM)
OPENIP/STARTIP	No equivalent command
OPENPORT/STARTPT	UPDATE PORT(name1,name2,...) START(COMM)
RECORDER OPEN RECORDER CLOSE	UPDATE MEMBER TYPE(IMSCON) START(TRACE) UPDATE MEMBER TYPE (IMSCON) STOP(TRACE)
SETRACF ON SETRACF OFF	UPDATE MEMBER TYPE(IMSCON) SET (RACF(ON)) UPDATE MEMBER TYPE(IMSCON) SET (RACF(OFF))
SETRRS	No equivalent command
STOPCLNT	DELETE PORT NAME(port) CLIENT(client1,client2,...)
STOPDS	UPDATE DATASTORE (name1,name2,...) STOP(COMM)
STOPIP	No equivalent command
STOPPORT	UPDATE PORT(name1,name2,...) STOP(COMM)
VIEWDS	QUERY DATASTORE NAME(name1,name2,...) [SHOW(ALL)]
VIEWHWS	QUERY MEMBER TYPE(IMSCON) [SHOW(ALL)]

REPLY command	MODIFY command
VIEWIP	No equivalent command
VIEWPORT	QUERY PORT NAME(port1,port2,...) [SHOW(ALL)]
VIEWUOR	QUERY UOR NAME(urid1,urid2,...) [SHOW(ALL)]

2.10.4 IMS command support for IMS Connect and OTMA

This section describes the IMS commands that can be used to check the status of IMS Connect and its interfacing OTMA instance.

/DISPLAY OTMA

The **/DISPLAY OTMA** command displays the XCF information, status, and security level. Example 2-27 shows an example of the output of this command.

Example 2-27 Output of the IMS /DISPLAY OTMA command

```

135 DFS996I *IMS READY* I13B
R 135,/DISPLAY OTMA
DFS000I  GROUP/MEMBER      XCF-STATUS  USER-STATUS  SECURITY  TIB  INPT SMEM  I13B
DFS000I                      DRUEXIT  T/O TPCNT ACEEAGE                      I13B
DFS000I                      I13B
DFS000I  -IVPAPLL1         NOT DEFINED  SERVER      FULL      0 10000      I13B
DFS000I  -IVPAPLL1         N/A          0          0                      I13B
DFS000I  *2013205/190047*  I13B

```

DISPLAY TMEMBER tmember_name Tpipe tpipe_ID

The **/DISPLAY TMEMBER** command can be used to display the status of the current transaction member status for OTMA clients and servers.

Example 2-28 shows all the members in the XCF group, which is similar to the **/DIS OTMA** command output.

Example 2-28 Output of the /DISPLAY TMEMBER ALL command

```

R 148,/DISPLAY TMEMBER ALL.
DFS000I  GROUP/MEMBER      XCF-STATUS  USER-STATUS  SECURITY  TIB  INPT SMEM  I13B
DFS000I                      DRUEXIT  T/O TPCNT ACEEAGE                      I13B
DFS000I  IVPAPLL1         NOT DEFINED  SERVER      FULL      0 10000      I13B
DFS000I  IVPAPLL1         N/A          0          0                      I13B
DFS000I  *2013205/193813*  I13B

```

A **/DISPLAY TMEMBER xxxx Tpipe ALL** command provides more information if there are active Tpipes.

2.11 IMS Connect operations and command enhancements

With the introduction of Open Database and SPOC, there are major enhancements in the arena of IMS Connect commands that allow you to monitor and control the IMS Connect environment effectively.

2.11.1 IMS Version 11: Enhanced IMS Connect commands

Several new and changed IMS Connect commands are available, mostly to support Open Database. Using these commands, you can perform the following actions:

- ▶ Query, stop, and start connections between IMS Connect and ODBM.
- ▶ Query, stop, and start ODBM alias names.

At the IMS Version 11 level, most commands have two formats: WTOR and z/OS MODIFY.

Value: The type-2 commands that are entered through the SPOC are an effective way to monitor and control IMSplex environments. IMS Connect now provides type-2 commands for the Open Database environment.

2.11.2 IMS Version 11: Enforcement of the single port requirement for SSL sockets

Before IMS Version 11, if users specified more than a single SSL port for an instance of IMS Connect, a failure could occur in the IBM Language Environment for z/OS. In IMS Version 11, IMS checks for the correct specification of SSL ports. If multiple SSL ports are specified through the **SSLPORT=** parameter, or if the SSL port specification exceeds eight characters, IMS Connect issues a message HWSX0909E and abends with a BPE abend code of U3401.

If users need multiple SSL ports for their IMS Connect connections, they can transfer the management of the SSL ports from IMS Connect to the IBM z/OS Communications Server Application Transparent Transport Layer Security (AT-TLS).

Value: This facility removes a way for an unnecessary IMS Connect abend to occur.

2.11.3 IMS Version 12: IMS Connect type-2 SPOC commands

IMS Connect is enhanced in IMS Version 12 to support a new type-2 command interface from the IMS SPOC. The following groups of commands are added:

- ▶ Display commands
- ▶ Start commands
- ▶ Stop and close commands
- ▶ Set, reset, and refresh commands

Display commands

Table 2-3 shows the new IMS Connect type-2 SPOC commands and the equivalent WTOR reply or type-2 modify commands that are related to displaying resources.

Table 2-3 IMS type-2 Display commands

Type-2 command	WTOR reply	Modify command
QUERY IMSCON TYPE(ALIAS) NAME(*) SHOW(ALL showparm)	VIEWIA ALL	Not applicable
QUERY IMSCON TYPE(ALIAS) NAME(alias) SHOW(ALL showparm)	VIEWIA alias	Not applicable
QUERY IMSCON TYPE(ALIAS) NAME(alias) ODBM(odbmname)	VIEWIA alias odbmname	Not applicable
QUERY IMSCON TYPE(CONFIG) SHOW(ALL showparm)	VIEWHWS	QUERY MEMBER TYPE(IMSCON) SHOW(ALL)
QUERY IMSCON TYPE(DATASTORE) NAME(*) SHOW(ALL showparm)	VIEWDS ALL	QUERY DATASTORE NAME(*) SHOW(ALL)
QUERY IMSCON TYPE(DATASTORE) NAME(datastore) SHOW(ALL showparm)	VIEWDS datastore	QUERY DATASTORE NAME(datastore)
QUERY IMSCON TYPE(IMSPLEX) NAME(*) SHOW(ALL showparm)	VIEWIP ALL	Not applicable
QUERY IMSCON TYPE(IMSPLEX) NAME(IMSplexname) SHOW(ALL showparm)	VIEWIP IMSplexname	Not applicable
QUERY IMSCON TYPE(MSC) NAME(*) SHOW(ALL showparm)	VIEWMSC ALL	QUERY MSC NAME(*)
QUERY IMSCON TYPE(MSC) NAME(mscid) SHOW(ALL showparm)	VIEWMSC mscid	QUERY MSC NAME(mscid)
QUERY IMSCON TYPE(PORT) NAME(*) SHOW(ALL showparm)	VIEWPORT ALL	QUERY PORT NAME(*) SHOW(ALL)
QUERY IMSCON TYPE(PORT) NAME(portname) SHOW(ALL showparm)	VIEWPORT portname	QUERY PORT NAME(portname) SHOW(ALL)
QUERY IMSCON TYPE(PORT) NAME(LOCAL) SHOW(ALL showparm)	VIEWPORT LOCAL	QUERY PORT NAME(LOCAL) SHOW(ALL)
QUERY IMSCON TYPE(RMTIMSCON) NAME(*) SHOW(ALL showparm)	VIEWRMT ALL	QUERY RMTIMSCON NAME(*)
QUERY IMSCON TYPE(RMTIMSCON) NAME(rmtimscon) SHOW(ALL showparm)	VIEWRMT rmtimscon	QUERY RMTIMSCON NAME(rmtimscon)
QUERY IMSCON TYPE(UOR) NAME(*) SHOW(ALL showparm)	VIEWUOR ALL	QUERY UOR NAME(*) SHOW(ALL)
QUERY IMSCON TYPE(UOR) NAME(uorid) SHOW(ALL showparm)	VIEWUOR uorid	QUERY UOR NAME(uorid) SHOW(ALL)

Figure 2-13 presents an example of a type-2 **QUERY IMSCON TYPE(PORT)** command that is issued from the SPOC.

PLX13		IMS Single Point of Control	
Command ==> _____			
_____		Plex . . _____	Route . . _____
_____		Wait . . _____	
Response for: QUERY IMSCON TYPE (PORT)			
Port	MbrName	CC	
6200D	HWSI13B1	0	
6201D	HWSI13B1	0	
7200	HWSI13B1	0	
7201	HWSI13B1	0	
7202	HWSI13B1	0	

Figure 2-13 Output of the type-2 **QUERY IMSCON TYPE(PORT)** command from the SPOC

Start commands

Table 2-4 shows the new IMS Connect type-2 SPOC commands and the equivalent WTOR reply or type-2 modify commands that are related to starting resources.

Table 2-4 IMS Connect type-2 start commands

Type-2 command	WTOR reply	Modify command
UPDATE IMSCON TYPE(ALIAS) NAME(alias) ODBM(odbmname) START(COMM)	STARTIA alias odbmname	Not applicable
UPDATE IMSCON TYPE(CONFIG) START(RECORDER)	RECORDER OPEN	UPDATE MEMBER TYPE(IMSCON) START(TRACE)
UPDATE IMSCON TYPE(DATASTORE) NAME(datastore) START(COMM)	OPENDS datastore or STARTDS datastore	UPDATE DATASTORE NAME(datastore) START(COMM)
UPDATE IMSCON TYPE(IMSPLEX) NAME(IMSpIexname) START(COMM)	OPENIP IMSpIexname or STARTIP IMSpIexname	UPDATE IMSPLEX NAME(IMSpIexname) START(COMM)
UPDATE IMSCON TYPE(MSC) NAME(1c1p1kid) START(COMM)	STARTMSC 1c1p1kid	UPDATE MSC NAME(1c1p1kid) START(COMM)
UPDATE IMSCON TYPE(ODBM) NAME(odbmname) START(COMM)	STARTOD odbmname	Not applicable
UPDATE IMSCON TYPE(RMTIMSCON) NAME(rmtimscon) START(COMM)	STARTRMT rmtimscon	UPDATE RMTIMSCON NAME(rmtimscon) START(COMM)
UPDATE IMSCON TYPE(PORT) NAME(portname) START(COMM)	OPENPORT portname or STARTPT portname	UPDATE PORT NAME(portname) START(COMM)

Stop and close commands

Table 2-5 on page 59 shows the new IMS Connect type-2 SPOC commands and the equivalent WTOR reply or type-2 modify commands that are associated with stopping or closing resources.

Table 2-5 IMS Connect type-2 stop or close commands

Type-2 command	WTOR reply	Modify command
UPDATE IMSCON TYPE(ALIAS) NAME(alias) ODBM(odbmname) STOP(COMM)	STOPIA alias odbmname	Not applicable
UPDATE IMSCON TYPE(CLIENT) NAME(client_name) PORT(portname) STOP(COMM)	STOPCLNT portname clientid	DELETE PORT NAME(portName) CLIENT(clientName)
UPDATE IMSCON TYPE(CONFIG) SHUTDOWN(COMM)	CLOSEHWS	SHUTDOWN MEMBER
UPDATE IMSCON TYPE(CONFIG) SHUTDOWN(COMM) OPTION(FORCE)	CLOSEHWS FORCE	SHUTDOWN MEMBER OPTION(FORCE)
UPDATE IMSCON TYPE(CONFIG) SHUTDOWN(COMM) OPTION(QUIESCE)	CLOSEHWS QUIESCE	SHUTDOWN MEMBER OPTION(QUIESCE)
UPDATE IMSCON TYPE(DATASTORE) NAME(datastore) STOP(COMM)	STOPDS datastore	UPDATE DATASTORE NAME(datastore) STOP(COMM)
UPDATE IMSCON TYPE(IMSPLEX) NAME(IMSplexname) STOP(COMM)	STOPIP IMSplexname	UPDATE IMSPLEX NAME(IMSplexname) STOP(COMM)
UPDATE IMSCON TYPE(LINK) NAME(linkname) STOP(COMM)	STOPLINK linkname	DELETE LINK NAME(linkname)
UPDATE IMSCON TYPE(MSC) NAME(1clp1kid) STOP(COMM)	STOPMSC 1clp1kid	UPDATE MSC NAME(1clp1kid) STOP(COMM)
UPDATE IMSCON TYPE(ODBM) NAME(odbmname) STOP(COMM)	STOPOD odbmname	Not applicable
UPDATE IMSCON TYPE(RMTIMSCON) NAME(rmtimscon) STOP(COMM)	STOPRMT rmtimscon	UPDATE RMTIMSCON NAME(rmtimscon) STOP(COMM)
UPDATE IMSCON TYPE(SENDCLNT) NAME(sendclient_name) RMTIMSCON(rmtimscon) STOP(COMM)	STOPSCLN rmtimscon sendclient	DELETE RMTIMSCON NAME(rmtimscon) SENDCLNT(clientid)
UPDATE IMSCON TYPE(PORT) NAME(portname) STOP(COMM)	STOPPORT portname	UPDATE PORT NAME(portname) STOP(COMM)
UPDATE IMSCON TYPE(CONFIG) STOP(RECORDER)	RECORDER CLOSE	UPDATE MEMBER TYPE(IMSCON) STOP(TRACE)

Set, reset, and refresh commands

Table 2-6 shows the new IMS Connect type-2 SPOC commands and the equivalent WTOR reply or type-2 modify commands that are associated with setting, resetting or refreshing resources.

Table 2-6 IMS Connect type-2 set, reset, and refresh commands

Type-2 command	WTOR reply	Modify command
UPDATE IMSCON TYPE(CONFIG) SET(OAUTO(ON))	SETOAUTO YES	UPDATE MEMBER TYPE(IMSCON) SET(OAUTO(ON))
UPDATE IMSCON TYPE(CONFIG) SET(OAUTO(OFF))	SETOAUTO NO	UPDATE MEMBER TYPE(IMSCON) SET(OAUTO(OFF))
UPDATE IMSCON TYPE(CONFIG) SET(PSWDMC(ON))	SETPWMC ON	UPDATE MEMBER TYPE(IMSCON) SET(PSWDMC(ON))

Type-2 command	WTOR reply	Modify command
UPDATE IMSCON TYPE(CONFIG) SET(PSWDMC(OFF))	SETPWMC OFF	UPDATE MEMBER TYPE(IMSCON) SET(PSWDMC(OFF))
UPDATE IMSCON TYPE(CONFIG) SET(PSWDMC(RCF))	SETPWMC RCF	UPDATE MEMBER TYPE(IMSCON) SET(PSWDMC(RCF))
UPDATE IMSCON TYPE(CONFIG) SET(RACF(ON))	SETRACF ON	UPDATE MEMBER TYPE(IMSCON) SET(RACF(ON))
UPDATE IMSCON TYPE(CONFIG) SET(RACF(OFF))	SETRACF OFF	UPDATE MEMBER TYPE(IMSCON) SET(RACF(OFF))
UPDATE IMSCON TYPE(CONFIG) SET(RRS(ON))	SETRRS ON	Not applicable
UPDATE IMSCON TYPE(CONFIG) SET(RRS(OFF))	SETRRS OFF	Not applicable
UPDATE IMSCON TYPE(CONFIG) SET(UIDCACHE(ON))	SETUIDC ON	UPDATE MEMBER TYPE(IMSCON) SET(UIDCACHE(ON))
UPDATE IMSCON TYPE(CONFIG) SET(UIDCACHE(OFF))	SETUIDC OFF	UPDATE MEMBER TYPE(IMSCON) SET(UIDCACHE(OFF))
UPDATE IMSCON TYPE(CONVERTER) NAME(converter_name) OPTION(REFRESH)	REFRESH CONVERTER NAME(converter_name)	UPDATE CONVERTER NAME(converter_name) OPTION(REFRESH)
UPDATE IMSCON TYPE(RACFUID) NAME(userid) OPTION(REFRESH)	REFRESH RACFUID NAME(userid)	UPDATE RACFUID NAME(userid) OPTION(REFRESH)

Value: This extends the benefits of IMS type-2 commands, and their associated improved ease of use for IMS Connect.

2.11.4 IMS Version 12: Partial read status display and control

Partial read is a TCP/IP status where the client sending data has not sent the whole message that the listener is expecting. This condition can lead to unexpected hangs in IMS Connect or in the client process.

IMS Connect in IMS Version 12 now has a function to deal with clients that have not completed sending their data or have the wrong value for the length that is passed to the IMS Connect exit. You can see which clients are waiting in a partial read status mode.

Example 2-29 shows the output from a **VIEWPORT** command with a client using HWSSMPL1 that has an invalid full length value (LLLL). The STATUS of READ indicates that we are in a partial read state. IMS Connect is reading a message from the client but has not yet received the entire length of the message. The client is hanging waiting for IMS Connect and IMS Connect is waiting for the client to send the missing data.

Example 2-29 Output from the VIEWPORT command with a partial read client

```
R 822,VIEWPORT 7400
IEE600I REPLY TO 822 IS;VIEWPORT 7400
HWSC0001I PORT=7400 STATUS=ACTIVE KEEPAV=0 NUMSOC=2 EDIT=
TIMEOUT=0
HWSC0001I CLIENTID USERID TRANCOD DATASTORE STATUS
SECOND CLNTPORT IP-ADDRESS APSB-TOKEN
HWSC0001I DELDUMMY READ
386 9180 127.000.000.001
```

```
HWSC0001I TOTAL CLIENTS=1 RECV=0 READ=1 CONN=0 XMIT=0 OTHER=0
```

Example 2-30 shows usage of the **STOPCLNT** command to terminate the connection from the remote client and break the hang condition.

Example 2-30 The STOPCLNT command

```
R 832,STOPCLNT 7400 DELDUMMY
IEE600I REPLY TO 832 IS;STOPCLNT 7400 DELDUMMY
HWSS0761I TCPIP COMMUNICATION WITH CLIENT=7400 _DELDUMMY STOPPED;
M=SCCM
```

Value: You can now determine which clients are waiting with a partial read status and take appropriate action, such as the **STOPCLNT** command, to break the hang condition.

2.11.5 IMS Version 13: IMS Connect command enhancements

Before IMS Version 13, when IMS Connect configuration definitions needed to be changed, the user had to shut down IMS Connect and restart IMS Connect for the changes to take effect. This reduces the availability of IMS Connect.

As described in 2.11.3, “IMS Version 12: IMS Connect type-2 SPOC commands” on page 56, both **QUERY** and **UPDATE** type-2 SPOC commands are available. Now, IMS Connect command enhancements provide two new and enhanced type-2 commands to create configuration definitions during IMS Connect runtime periods.

Like all type-2 commands, these type-2 commands are supported only through the Operations Manager (OM) API, such as TSO SPOC, REXX SPOC API, Batch SPOC, and the IMS Control Center.

The following new type-2 commands for IMS Connect were introduced:

- ▶ **CREATE IMSCON TYPE(DATASTORE)**

Use the **CREATE IMSCON TYPE(DATASTORE)** command to add a communications path to an IMS datastore from IMS Connect.

- ▶ **CREATE IMSCON TYPE(PORT)**

Use the **CREATE IMSCON TYPE(PORT)** command to create a listening port in IMS Connect.

Value: For DATASTORE and PORT configuration definition changes, you no longer must recycle IMS Connect.

2.11.6 IMS Version 13: Increasing the maximum number of XML converters

The IMS Connect XML Adapter uses COBOL Converters to perform transformations. The converters are generated from IBM Rational® Developer for System z® tools and loaded into the IMS Connect address space. Currently, a maximum of 100 XML converters can be loaded in IMS Connect.

This enhancement adds an IMS Connect configuration parameter **MAXCONV** in the **ADAPTER** statement, which increases the maximum number of XML converters that can be loaded by this instance of IMS Connect. Also, an IMS Connect type-2 command **QUERY IMSCON TYPE(CONVERTER)** is available to query the XML converters that are loaded in IMS Connect.

Value: Increases the scalability of Rational Developer for System z tools that create XML converters in the IMS Connect address space.

2.11.7 IMS Version 13: Auto-restarting the Language Environment

IMS Connect uses the Language Environment and the IMS Base Primitive Environment (BPE) to run the XML converters when requested by IMS SOAP Gateway. Before IMS Version 13, when one of the XML converters abnormally ends (ABENDs), it cannot be reused, but an issue with Language Environment causes IMS Connect to run the bad XML converter a total of three times before the Language Environment can be successfully terminated. To restart the Language Environment environment, IMS Connect must be restarted.

This IMS Version 13 enhancement provides an automated mechanism for restarting the Language Environment after one of the XML converters ABENDs, which means that previously loaded converters are reloaded the next time that they must be used.

IMS Connect also automatically makes a request to BPE to refresh the XMLADAP BPE User Exit after the ABEND limit (ABLIM), which specified in BPE's **EXITDEF()** statement for HWS USER EXIT DEFINITION, has been reached. This auto refresh allows IMS Connect users to keep IMS Connect running without having to manually refresh the XMLADAP BPE User Exit for IMS Connect after the ABEND limit is reached.

Value: The benefit of this facility is to improve efficiencies during error conditions and also eliminate unnecessary IMS Connect restarts when this situation occurs.

2.12 Accessibility through IMS Connect

IMS Connect can either be on the same z/OS image as its target IMS or can cross a system boundary (assuming that the environment is sysplex-enabled). Connection requests to IMS Connect can be short-lived, which is the case for transaction sockets and non-persistent sockets, or they can be long-lived, which is the case for persistent sockets.

When IMS Connect is provided with access to multiple IMS systems, code can be added to the IMS Connect user message exits to perform load balancing and failover. The code to do this must be user-written unless the environment includes a product such as IMS Connect Extension, which provides an interface that facilitates this capability.

2.13 IMS Connect accessibility enhancements

This section lists the accessibility enhancements that are provided for IMS Connect to connect to data store managers, and other IMS systems through TCP/IP connectivity, by MSC using TCP/IP, and CICS ISC usage of TCP/IP to interface with IMS.

2.13.1 IMS Version 11: Integrated IMS Connect enhancements for IMS DB

The enhancements to IMS Connect for IMS DB include a new IBM Distributed Relational Database Architecture™ (DRDA) compliant application programming interface (API) and the ability to communicate with the new IMS Open Database Manager address space.

With IMS Version 11, IMS Connect is the TCP/IP path into IMS DB and IMS TM. IMS Connect clients access IMS DB through a new API, one that adheres to the open DRDA standard specifications and supports Distributed Data Management (DDM) architecture commands.

As part of the new Open Database enhancements, IMS Version 11 provides new Java drivers, called the IMS Universal drivers, that you can use to access your IMS data. IMS Connect supports the IMS Universal drivers with the DRDA API. Independent software vendors can also use any of the IMS Universal drivers to build packages that access IMS data.

Value: IMS Open Database provides distributed access to IMS database resources.

2.13.2 IMS to IMS TCP/IP connectivity

IMS Version 12 can send messages from a local IMS to a remote IMS over a Internet Protocol network. The messages are sent over the Internet Protocol network by a local IMS Connect and received by a remote IMS Connect. The local IMS communicates with the local IMS Connect and the remote IMS communicates with the remote IMS Connect. IMS can send messages using OTMA or MSC protocols.

Existing methods in use before IMS Version 12 typically require the creation of customer gateway applications that implement Resume Tpipe code to retrieve the message from one IMS along with code to establish a connection and send the message to another IMS system.

With this new capability, IMS and IMS Connect provide the necessary communication path and code that eliminate the need for a gateway application. For OTMA messages, one-way asynchronous messaging is supported. IMS OTMA communicates with IMS Connect using XCF. Figure 2-14 presents this concept.

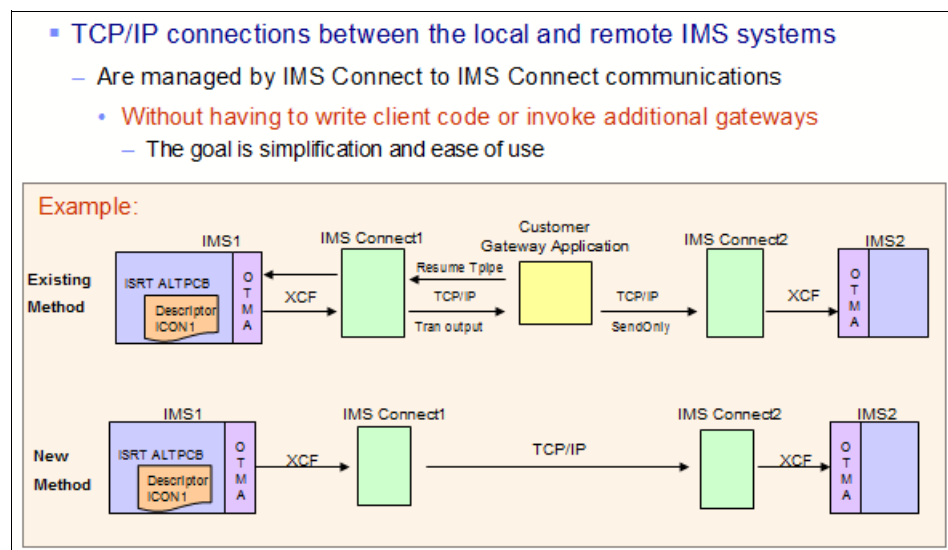


Figure 2-14 Asynchronous IMS to IMS communication through TCP/IP

Figure 2-15 provides more detail about asynchronous IMS to IMS communication through TCP/IP.

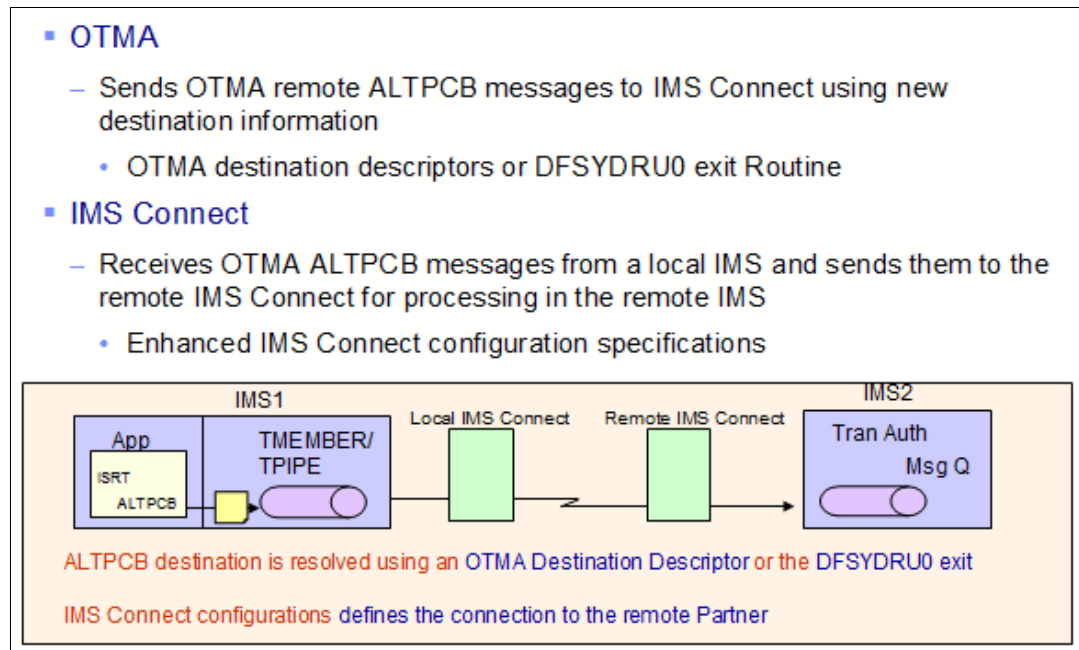


Figure 2-15 Asynchronous IMS to IMS communication through TCP/IP details

The following actions take place:

1. The IMS application in IMS1 makes an ISRT ALTPCB call. If an OTMA descriptor is used, then the application specifies a descriptor name as the destination on the change call. IMS1/OTMA looks up the descriptor name to retrieve the values that describe the remote connection and inserts them into the OTMA headers of the message. IMS1/OTMA then sends the message to the local IMS Connect specified in the **TMEMBER** parameter of the descriptor. If the OTMA descriptor is not specified, the OTMA DFSYDRU0 user exit can be coded to provide the appropriate values for the remote connection.
2. IMS1 waits for an ACK.
3. IMS Connect1 receives the message, removes the OTMA headers, and builds an IRM and sends the message to the remote IMS Connect that is specified in the OTMA headers using SendOnly with ACK. IMS Connect1 waits for an ACK.
4. IMS Connect2 receives the message and sends the message to the remote IMS.
5. When the remote IMS2 receives the message and queues the message, an ACK is sent back to IMS Connect2. IMS Connect2 sends the ACK back to IMS Connect1, which sends the ACK back to IMS1. IMS1 dequeues the message from its Tpipe.
6. The remote IMS processes the message as a transaction. Responses, if any, from the IMS application are sent to the remote IMS Connect Tpipe.

There are four new parameters that were added to the OTMA destination routing descriptor in support of IMS Connect to IMS Connect connectivity: **RMTIMSCON**, **RMTIMS**, **RMTTRAN**, and **USERID**.

- ▶ **RMTIMSCON** is the name of the remote IMS Connect connection to be used by the local IMS Connect. This value must match a value in the **ID** parameter of the **RMTIMSCON** statement in the local IMS Connect. If this parameter is specified, the parameter **RMTIMS** must also be specified.
- ▶ **RMTIMS** is the name of the remote IMS to which the message is sent. This value is the same value that is specified in the **ID** parameter of the **DATASTORE** statement in the remote IMS Connect. If this parameter is specified, the parameter **RMTIMSCON** must also be specified.
- ▶ **RMTTRAN** is optional and defines the transaction code to use at the remote IMS. If a value is not specified, the transaction in the start of the message is used.
- ▶ **USERID** is optional and is the SAF (RACF) user ID name to use at the remote IMS. If this parameter is not specified, the user ID from the IMS application that did the ISRT is used.

The IMS Connect definition file in Example 2-10 on page 43 contains the **RMTIMSCON** and **RMTIMS** parameters.

Alternatively, the user exit DFSYDRU0 can provide the connection values. A sample DFSYDRU0 exit provides details about the output flag bit indicator in the output parameter list.

Value: IMS and IMS Connect provide the necessary TCP/IP communication path and code that eliminate the need for a gateway application.

2.13.3 IMS Version 12: Multiple Systems Coupling using TCP/IP

IMS has had MSC using **TYPE=CTC**, **TYPE=MTM**, and **TYPE=VTAM** for many releases. The support in IMS Version 12 adds a **TYPE** to the **MSPLINK** macro. You can now define an **MSPLINK** with **TYPE=TCPIP**.

Figure 2-16 shows a general overview of how the TCP/IP MSC link support in IMS Version 12 is configured.

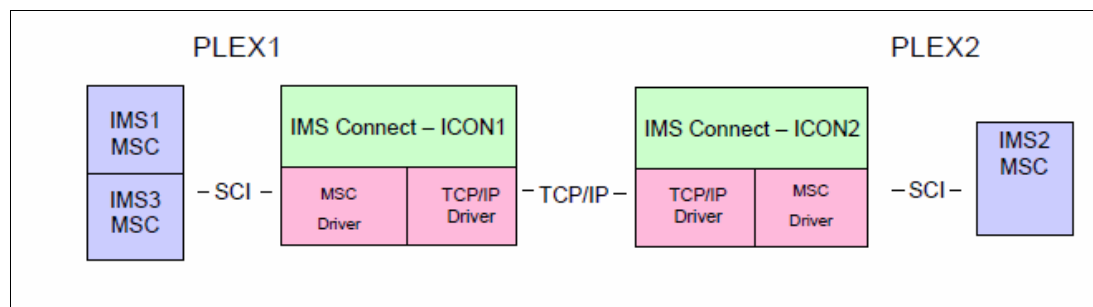


Figure 2-16 MSC using TCP/IP for IMS Connectivity

Stage 1 definitions that are needed in IMS

The new TCP/IP MSC links must be defined in IMS in the stage1 or stage2 system generation in the same way that you define a **TYPE=VTAM**, **TYPE=CTC**, or **TYPE=MTM** link. Only the **MSPLINK** macro has changed. The other stage1 macros to define the **MSLINK** and **MSNAME** remain the same.

Example 2-31 shows the IMS stage1 macros that are needed to define an MSC link using TCP/IP for the local IMS system. The MSLINK and MSNAME macros have no special requirements.

Example 2-31 IMS stage 1 macros for a TCP/IP MSC link (local system)

```
LINKBA MSPLINK TYPE=TCPIP,NAME=I12B,SESSION=2,BUFSIZE=4096, X
LCLICON=HWSI12A1, X
LCLPLKID=MSC2A2B
MSLINK PARTNER=BA,MSPLINK=LINKBA
MSCBA MSNAME SYSID=(102,101)
```

The value in the **LCLICON** parameter is also specified on the **MEMBER** parameter of the **IMSPLEX** substatement of the MSC configuration statement for the IMS Connect instance.

The value in the **LCLPLKID** parameter equals the ID of the MSC configuration statement that defines the MSC physical link to the IMS Connect instance. This value must match the value that is specified on the **LCLPLKID** keyword of the MSC configuration statement for this IMS Connect instance.

Example 2-32 shows the definitions for the remote IMS system.

Example 2-32 IMS stage 1 macros for a TCP/IP MSC link (remote system)

```
LINKBA MSPLINK TYPE=TCPIP,NAME=I12A,SESSION=2,BUFSIZE=4096, X
LCLICON=HWSI12B1, X
LCLPLKID=MSC2B2A
MSLINK PARTNER=BA,MSPLINK=LINKBA
MSCBA MSNAME SYSID=(101,102)
```

Consideration: When the first **TYPE=TCPIP MSPLINK** is added to your IMS stage1, you must run an ALL or NUCLEUS system generation and restart IMS.

IMS Connect definitions

The new TCP/IP MSC links require two more configurations in IMS Connect: An **MSC** statement to define the local **MSPLINK**, and an **RMTIMSCON** statement to define the remote IMS Connect system. Example 2-33 shows the local definition that is required in IMS Connect.

Example 2-33 IMS Connect configuration for a TCP/IP MSC link (local system)

```
HWS=(ID=HWSI12A1,PSWDMC=N,RACF=N,RRS=Y,UIDCACHE=Y,XIBAREA=50)
TCPIP=(EXIT=(HWSSMPL1,HWSOAP1),HOSTNAME=TCPIP,PORT(ID=7102))
MSC=(LCLPLKID=MSC2A2B,RMTPLKID=MSC2B2A,
LCLIMS=I12A,RMTIMS=I12B,
IMSPLEX=(MEMBER=HWSI12A1,TMEMBER=IM12X),
RMTIMSCON=HWSI12B1)
RMTIMSCON=(ID=HWSI12B1,
HOSTNAME=WTSC64.ITS0.IBM.COM,PORT=7202,
PERSISTENT=Y,RESVSOC=1)
```

Example 2-34 shows the remote definition that is required in IMS Connect.

Example 2-34 IMS Connect configuration for a TCP/IP MSC link (remote system)

```
HWS=(ID=HWSI12B1,PSWDMC=N,RACF=N,RRS=Y,UIDCACHE=Y,XIBAREA=50)
TCPIP=(EXIT=(HWSSMPL1,HWSOAP1),HOSTNAME=TCPIP,PORT(ID=7202))
MSC=(LCLPLKID=MSC2B2A,RMTPLKID=MSC2A2B,
```

```
LCLIMS=I12B,RMTIMS=I12A,  
IMSPLEX=(MEMBER=HWSI12B1,TMEMBER=IM12X),  
RMTIMSCON=HWSI12A1)  
RMTIMSCON=(ID=HWSI12A1,  
HOSTNAME=WTSC63.ITS0.IBM.COM,PORT=7102,  
PERSISTENT=Y,RESVSOC=1)
```

The relationship between the **MSC** and **RMTIMSCON** configuration statements in IMS Connect is created by the **RMTPLKID**, **RMTIMS**, **RMTIMSCON**, and **ID** specifications.

An IBM white paper, *IMS Version 12 Performance Evaluation Summary - Maintaining superior performance*, produced by the IMS laboratory in September 2011, published the results of comparison tests between MSC TCP/IP and MSC using VTAM. The IMS Version 12 TCP/IP MSC type link achieved 10,587 transactions per second compared to 9,303 transactions per second with VTAM MSC type links. with greater efficiency. This white paper can be found at:

http://public.dhe.ibm.com/software/data/sw-library/ims/IMS_Version_12_Performance_Summary.pdf

Value: IMS Connect provides routing and connection management support for MSC links that connect to a TCP/IP generic resource group. This support provides a TCP/IP solution for Multiple Systems Coupling interfaces, which can be used in a larger effort to move your entire IMS environment to the TCP/IP communication.

2.13.4 IMS Version 13: ISC using TCP/IP protocols

IMS Version 13 enhances Intersystem Communication (ISC) connectivity to include the TCP/IP protocol. This capability applies to connections between IMS and Customer Information Control System (CICS) systems. This new support provides a solution for environments that want to implement an all-inclusive Internet Protocol network environment.

The benefits are:

- ▶ You now have a strategic protocol alternative to SNA/VTAM.
- ▶ You can have an all-inclusive TCP/IP solution for your networks.

At a high level, the ISC support using TCP/IP protocols can be defined in IMS environments both statically and dynamically (ETO). The static definitions are the traditional type 1 sysgen (**TYPE**, **TERMINAL**, and **SUBPOOL**) macros. The dynamic terminal specifications are set through logon descriptors in DFSDSCMx or DFSDSCTy proclib members. It uses the structure call interface (SCI) of the common service layer (CSL) between IMS and IMS Connect to leverage the existing IMS Connect TCP/IP support.

ISC VTAM (SNA LU 6.1) connections continue to be supported in IMS Version 13 along with the new ISC TCP/IP capability. Connectivity to CICS using TCP/IP requires CICS Version 5.1.

For more information about the performance of IMS V13 with TCP/IP and CICS, see the IBM white paper *IMS 13 Performance Evaluation Summary - Reducing the Total Cost of Ownership with Improved Performance*, found at:

<http://public.dhe.ibm.com/common/ssi/ecm/en/iml14383usen/IML14383USEN.PDF>

Figure 2-17 presents an overview of the features of this new facility.

- **ISC communication over TCP/IP networks**
 - Supports both static and dynamic terminals
 - Static terminal definitions
 - SYSGEN stage 1 TYPE, TERMINAL, SUBPOOL macros
 - DFSDCxxx PROCLIB member
 - Dynamic terminal specification
 - Logon descriptors in the DFSDSCMx or DFSDSCTy PROCLIB member
 - Leverages IMS Connect
 - Enhancements to the HWSCFGxx configuration member
 - No IMS Connect user message exit changes
 - Uses CSL to communicate between IMS and IMS Connect
 - Requires Structured Call Interface (SCI) and Operation Manager (OM)
- **IVP and Syntax Checker enhancements**

Figure 2-17 Overview of the ISC communication solution over Internet Protocol networks

The only supported flow is asynchronous for both IMS and CICS origins.

Figure 2-18 presents support functionalities with IMS Version 13 and the current restrictions.

ISC communication over TCP/IP networks

Functionality – Support and Restrictions

Functions And Transactions	Existing in LU6.1	Supported in TCP/IP
CICS transaction – START/RETRIEVE	Yes	Yes
CICS transaction – SEND(INVITE)/RECV	Yes	No
CICS transaction – SEND(LAST)/RECV	Yes	No
IMS non-response mode transaction	Yes	Yes
IMS response mode transaction (including FP)	Yes	No
IMS conversational mode transaction	Yes	No
IMS recoverable transaction	Yes	Yes
IMS non-recoverable transaction	Yes	Yes
IMS message switch	Yes	Yes
IMS operator command	Yes	Yes
Dynamic terminal	Yes	Yes
Static terminal	Yes	Yes
Front-End Switch (FES)	Yes	No
Message Format Service (MFS)	Yes	No
IMSplex Terminal Management (STM)	Yes	No
VTAM Generic Resources (VGR)	Yes	No
Extended Recovery Facility (XRF)	Yes	No

Figure 2-18 ISC communication over TCP/IP functionality

There are two methods of initiating transactions by using a TCP/IP connection:

- ▶ Initiating a CICS transaction from IMS using an ISC TCP/IP connection
- ▶ Initiating an IMS transaction from CICS using an ISC TCP/IP connection

Figure 2-19 presents these two flows.

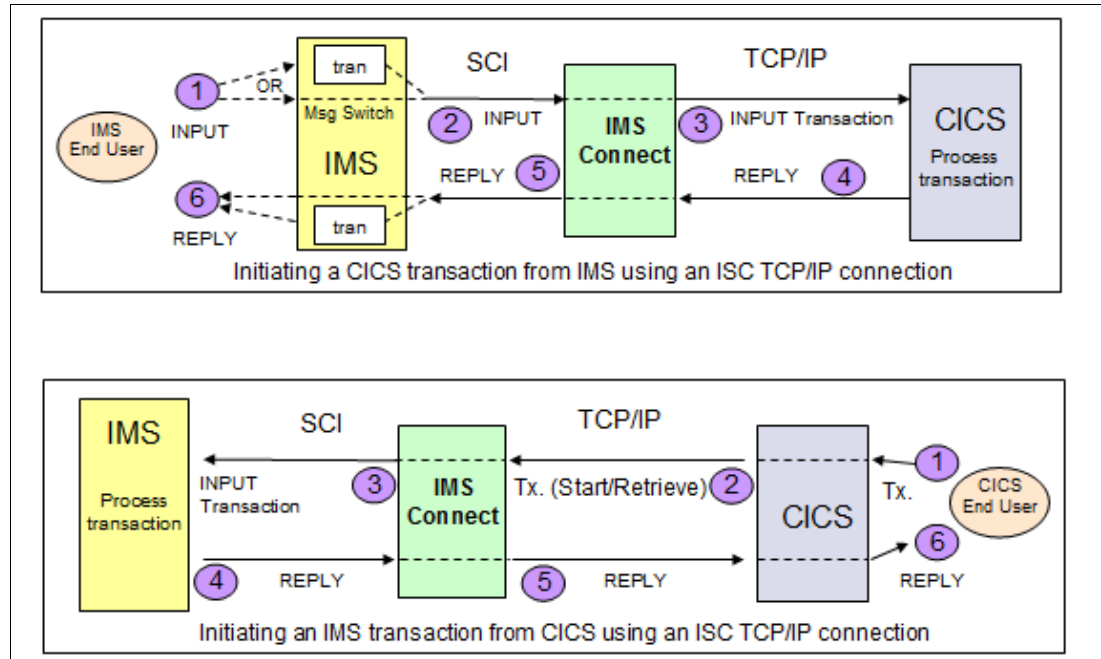


Figure 2-19 Initiating transactions using an ISC TCP/IP connection

Initiating a CICS transaction from IMS using an ISC TCP/IP connection as the following steps:

1. An IMS user sends a message-switch that is sent to CICS or an input message to an IMS transaction that issues **ISRT**, **ALTPCB** for CICS.
2. IMS uses an active ISC TCP/IP connection to pass the message to IMS Connect using SCI.
3. IMS Connect uses an active socket connection to send the message to CICS.
4. CICS receives the message as a transaction from the Internet Protocol network, processes it, and uses an active ISC TCP/IP connection to send the transaction reply back to IMS.
5. IMS Connect receives the transaction reply from the Internet Protocol network, and passes it to IMS over an active ISC TCP/IP connection using SCI.
6. The reply can start an IMS transaction or be a reply that is targeted for an IMS user.

Initiating an IMS transaction from CICS using an ISC TCP/IP connection has the following steps:

1. The CICS client enters a transaction request.
2. CICS uses an active ISC TCP/IP connection to send the transaction to IMS.
3. IMS Connect receives the transaction from the Internet Protocol network and uses an active ISC TCP/IP connection to pass the message to IMS using SCI.
4. IMS processes the transaction, and uses an active ISC TCP/IP connection using SCI to pass the transaction reply to IMS Connect.

5. IMS Connect uses an active socket connection to send a transaction reply to CICS.
6. CICS sends the reply back to the client.

The Operations Manager (OM) provides type-2 command support for the environment. The new ISC TCP/IP support does not impose any change on the existing CSL definition structure in the DFSDFxxx member of IMS PROCLIB. The **IMSPLEX** parameter value in DFSDFxxx is the same value that is referenced in the IMS Connect setup definitions for ISC.

Figure 2-20 presents this interface in pictorial form.

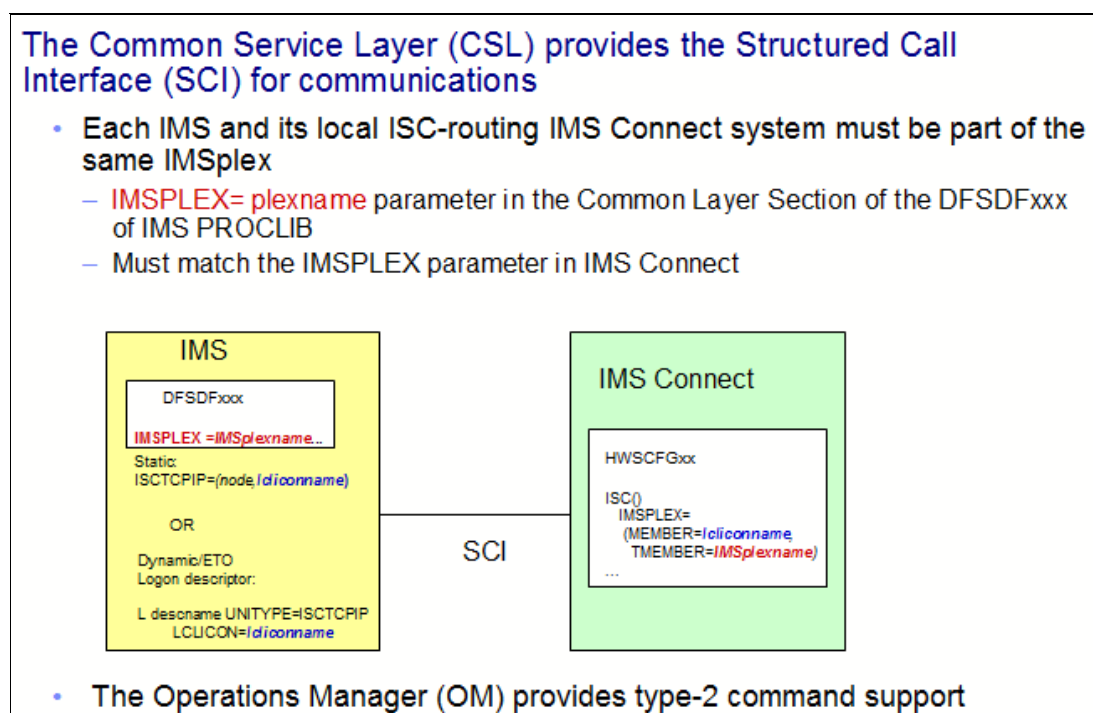


Figure 2-20 Usage of CSL in support of the IMS and IMS Connect ISC routing components

TCP/IP support for ISC is provided by IMS Connect and is defined by coding the appropriate configuration statements in the IMS Connect PROCLIB member. These statements include a new **CICSPORT** parameter in the existing TCPIP configuration statement, and new **RMTCICS** statements.

In the configuration that is shown in Figure 2-21 on page 71, note the RMTCICS definition. This definition shows that the local IMS Connect instance was defined to use port 8891 for outbound communications with CICS. 8891 is the CICS port that is used to listen for IMS Connect traffic, which is used for establishing the SEND sockets from IMS Connect.

■ HWSCFGXX Configuration sample

```

TCPIP=(HOSTNAME=TCPIP,PORTID(9996,9997,9998,9999),
      RACFID=GOFISHIN,MAXSOC=5000,
      CICSSPORT=(ID=9991),CICSSPORT=(ID=9992),CICSSPORT=(ID=9993),
      CICSSPORT=(ID=9994),CICSSPORT=(ID=9995),...)

RMTCICS (ID=CICS1,PORT=8891,
        HOSTNAME=HOSTB.COM,RESVSOC=10)

ISC (ID=ISC1,NODE=CICSA1,LCLIMS=IMS1,CICSSPORT=9991,
    RMTCICS=CICS1,CICSAPPL=CICSA1,CICSNETID=CICSNET,
    IMSPLEX=(MEMBER=ICON1,TMEMBER=PLEX1))

```

Figure 2-21 IMS Connect configuration sample that is associated with ISC support

The ISC statement shows that IMS Connect has specified that CICSSPORT 9991 is for inbound communication from CICS. This setup is for establishing the RECV sockets for messages from CICS. The CICSSPORT number matches one that was defined in the CICSSPORT parameter of the TCPIP statement.

As shown in Figure 2-22, IMS initiates the session by running /OPNDST on the IMS side. This configuration example shows two statically defined parallel sessions and subpools SSN1 and SSN2. Corresponding definitions are provided in CICS with IPCONN statements that match the subpools.

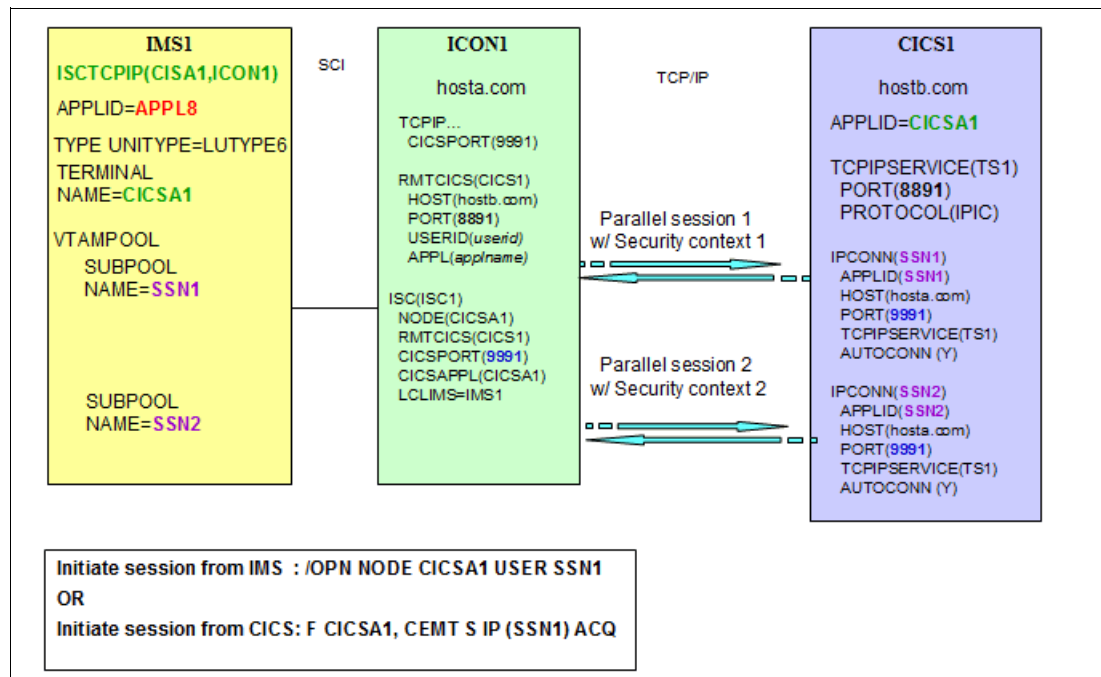


Figure 2-22 ISC TCP/IP component interface

IMS initiates the session by running `/OPNDST`. For example:

```
/OPN NODE CICSA1 USER SSN1
```

Alternatively, CICS can initiate the session by using the CEMT transaction. For example:

```
F CICSA1,CEMT S IP(SSN1) ACQ
```

New IMS Connect type-2 commands that are associated with ISC TCP/IP

Several commands were updated in support of ISC through IMS Connect.

- ▶ `QUERY IMSCON TYPE(ISC)`
- ▶ `QUERY IMSCON TYPE(ISCUSER)`
- ▶ `QUERY IMSCON TYPE(RMTCICS)`
- ▶ `UPDATE IMSCON TYPE(ISC) START|STOP(COMM)`
- ▶ `UPDATE IMSCON TYPE(RMTCICS) START|STOP(COMM)`
- ▶ `UPDATE IMSCON TYPE(ISCUSER) START|STOP(COMM)`

Value: This new support provides a solution for environments that want to implement an all-inclusive Internet Protocol network environment.

2.14 IMS Connect client programming interfaces

IMS Connect receives data from a TCP/IP client, performs basic editing and translation, starts security, and prepares the message to be in a format that is recognizable to OTMA. Response messages from IMS are also prepared to be in a format that the TCP/IP client understands.

IMS Connect communicates with OTMA through an XCF session by using the OTMA message headers. Clients that use TCP/IP socket calls as their communication vehicle can design a user exit routine that runs with IMS Connect to convert messages between the following formats:

- ▶ Converts the client message into an OTMA message format.
- ▶ Converts the IMS response in an OTMA message format to a client message format.

Figure 2-23 presents an overview of a message flow.

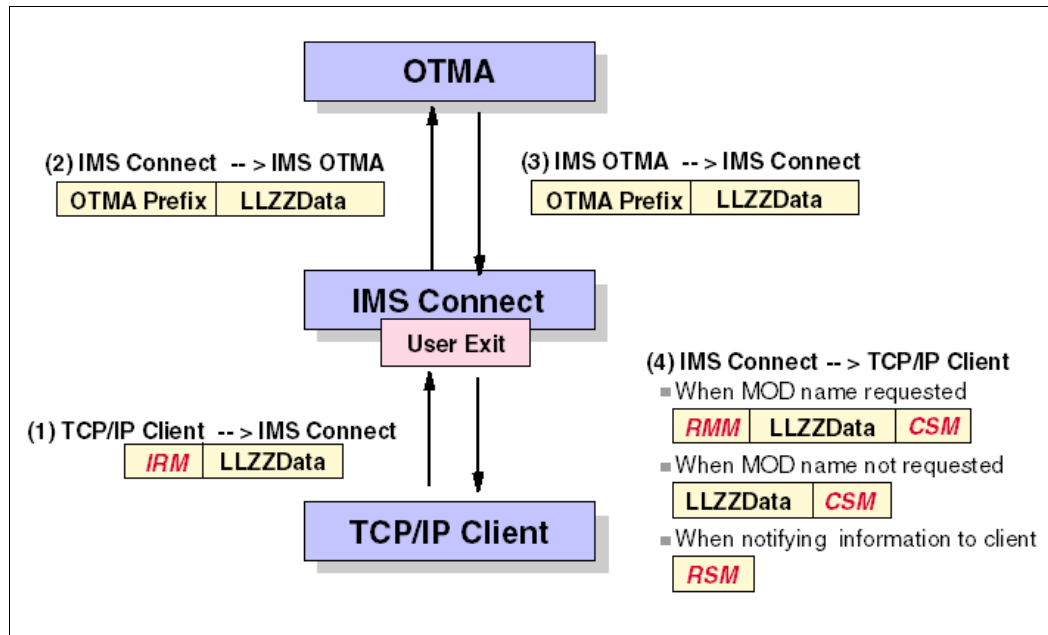


Figure 2-23 Message flow between a TCP/IP client and IMS (OTMA)

There are a few fields to define to understand this message flow:

► IMS Request Message (IRM)

IMS Connect expects all client messages that it receives to start with a 4-byte total length field, followed by an IMS Request Message (IRM) prefix. The TCP/IP client communicates with IMS by sending an IRM as the first message segment to IMS Connect. The input data stream consists of the IRM, immediately followed by all segments of application data.

► Request Status Message (RSM)

IMS Connect returns the Request Status Message (RSM) as the structure of an output message if IMS Connect or the message exit determines that an error occurred. The RSM contains a return and reason code indicating the type of status.

► Complete Status Message (CSM)

IMS Connect returns the Complete Status Message (CSM) as the last structure of an output message if the input message is processed successfully. It does not generate an end-of-message (EOM) segment.

► Request Mod Message (RMM)

IMS Connect returns the Request Mod Message (RMM) as the first structure of an output message, if the Message Format Service (MFS) message output descriptor (MOD) name is requested and the data output is present.

These conversions enable the client to retrieve IMS data through a TCP/IP connection. Therefore, clients that communicate with IMS Connect must follow the message structure that is predefined by the user exit routine.

2.15 IMS Connect Client programming interface enhancements

Client applications control much of the “rules of engagement” between themselves and IMS Connect at connectivity time. This section introduces a few enhancements.

2.15.1 IMS Version 10: Alternative client ID support

IMS Connect introduces a new protocol that allows client applications to specify an alternative client ID in the RESUME Tpipe request. IMS Connect forwards the alternative client ID to OTMA, and OTMA returns the asynchronous messages that are queued to the Tpipe of the alternative client ID name to the client application that issued the Resume Tpipe request.

Value: Allows Java application to use a shareable persistent socket to retrieve asynchronous output messages (RESUME Tpipe) from any Tpipe.

2.15.2 IMS Version 10: Send-only with acknowledgement protocol

IMS Connect is enhanced with a “send-only with acknowledgement” protocol. When the send-only with acknowledgement protocol option is specified, the client application receives an ACK response message from OTMA for each input message that is successfully enqueued by IMS. All other output that is generated by the send-only transaction is sent to the asynchronous hold queue.

IMS Connect is enhanced with another option of “send-only with serial delivery” by using the **DELIVERMSG=ORDERED** option of the XCF IXCMMSGO macro. When the send-only with serial delivery protocol option is specified, IMS Connect ensures that the order in which it submits send-only transactions to OTMA is in fact the order in which IMS receives the transactions.

Value: The ensured order of the submission of the send-only transaction aligns with the receive order.

2.15.3 Cancel Client ID support

The connection between a remote client application and IMS Connect sometimes is disrupted because TCP/IP failures, or processing failures on the client side. Because the original client status might still be active in IMS Connect, a request to reestablish the connection to the same port using the same client ID can result in a duplicate client condition and message HWSS0742W.

With this new function, the client application can establish a new connection with a request to “Cancel Client ID”, which causes IMS Connect to automatically discard and clean up any previously active session for that client ID. All CM0 or CM1 transactions running on either a persistent or transaction socket are supported.

Value: Allows more control of the session state between a client application and IMS Connect.

2.16 IMS Connect security

A security system must provide the following features:

- ▶ Authentication

The security system must be able to determine whether a client or a person is who or what it declares itself to be.

- ▶ Authorization

The security system must be able to determine whether a client or person has the right to perform an action or to access specific information.

- ▶ Secure communication

The security system must be able to ensure that the transmitted information can be seen only by the correct destination. TCP/IP connectivity offers multiple solutions to connect to a host system. No matter what type of TCP/IP client tries to connect to IMS, each client must perform the following actions:

- Prove their origin.
- Declare who they are.

2.16.1 IMS Connect secure access to IMS

The **USERID=&userid** parameter that is specified in the JOB card of the IMS Connect JCL is used as the security vehicle to ensure IMS Connect access to IMS. &userid must have READ access to the IMSXCF.group.member. IMS OTMA provides security for the IMS XCF connection by defining and permitting IMSXCF.group.member in the RACF FACILITY class.

To configure security for the local option using RACF: You must add HWS.ICON_NAME as the SAF facility class name (whether you configured security with the IMS Connect configuration member or **SETRACF** command). ICON_NAME is how IMS Connect is defined in the **ID** parameter of the **HWS** statement in the IMS Connect configuration member. The resource that must access IMS Connect is the WebSphere Application Server, and UPDATE authority is required to update the RACF profile.

2.16.2 Connecting IMS Connect to OTMA

Before IMS Connect can send any messages to IMS through OTMA for processing, IMS Connect must identify itself as an OTMA client. This task is achieved by issuing an OTMA command of the type client bid, passing the required security data to OTMA for verification. If your IMS is RACF-protected, the user ID passed by IMS Connect in the client bid must have READ access to the FACILITIES class entry `IMSXCF.xcf_group_name.xcf_member_name (client)` in RACF. If rejected by RACF, the client bid is denied and IMS Connect cannot connect to IMS as an OTMA client.

2.16.3 User verification

After IMS Connect successfully joins the XCF group and connects as an OTMA client to IMS, you can let IMS Connect do RACF user ID and password verification of each client on a per-message basis. This facility is driven by the **RACF=Y | N** parameter, as specified in the **HWSCFG** configuration file.

The user ID and password can be set up in one of two places:

- ▶ The originating client can build and send the security data as part of the message that is sent to IMS Connect through TCP/IP.
- ▶ The user message exit that gets driven after IMS Connect receives the complete message from the TCP/IP client.

2.16.4 IMS Connect SSL connections

To use Secure Sockets Layer (SSL) support with IMS Connect, you can either configure the IMS Connect support for SSL by using the IMS Connect SSL interface or by using the IBM z/OS Communications Server Application Transparent Transport Layer Security (AT-TLS) feature.

SSL and TLS protect the privacy and integrity of data that is transferred through a network. SSL rests on top of TCP/IP to provide a mechanism for secure sockets. SSL uses a combination of public and private keys and symmetric key encryption to authorize clients and servers to one another.

If you use AT-TLS, the usage of SSL is transparent to IMS Connect and you do not need to configure the IMS Connect SSL interface. Your z/OS TCP/IP administrator configures and administers SSL. For information about configuring AT-TLS, see *z/OS V1R13 Communications Server IP Configuration Guide*, SC31-8775-20.

In addition to simplifying IMS Connect security implementation, AT-TLS provides greater flexibility regarding the usage of ports. IMS Connect support for SSL is restricted to a single port. AT-TLS does not have this restriction.

Setting up IMS Connect SSL ports

As an optional task, you can set up Secure Socket Layer (SSL). On your z/OS system, verify that the SSL ports are active by running the IBM MVS™ Modify Command, Control Center, or MVS WTOR. For example, you can use the **VIEWHWS** command to display a list of the SSL ports and details their status. Ensure that some of the ports have the suffix **s** and that they are in the **ACTIVE** state.

For more information about how to set up the SSL configuration in IMS Connect, see the “IMS Connect SSL connections” section in *IMS V13 Communications and Connections*, SC19-3651-00.

2.17 IMS Connect security enhancements

IMS Connect has kept up with requirements to secure its environment as more business critical workloads flow through it.

IMS Version 10: RACF mixed case passwords

The support for a RACF mixed case password in IMS Connect is aligned with the IMS Version 10 support for mixed case passwords. This capability in IMS Connect allows the password to be preserved exactly as the remote client provided it, and passes the string to RACF without translation to uppercase.

To turn on this function, use parameter **PSWDMC=** in the **HWS=** statement, which defines the option of mixed case passwords. **PSWDMC=N** is the default. This setting can be changed by using the IMS Connect **SETPWMC** or **UPDATE** commands.

IMS Connect support requires that RACF enable mixed case passwords through the RACF **SETROPTS(MIXEDCASE)** command. The RACF enablement of this support does not constitute the IMS Connect usage of this support. Also, the mixed case support for IMS Connect can take effect only when RACF is enabled.

Value: The usage of mixed case password support is an ease-of-use facility for the remote client.

2.17.1 IMS Version 10: Client password change request

User exit routines HWSSMPL0, HWSSMPL1, and HWSJAVA0 can submit input messages to change the passwords that are used by IMS Connect client application programs. The routines check for a leading keyword of 'HWSPWCH' to determine whether it is a request to change the password. This 'HWSPWCH' string can be viewed as a transaction code, but new logic in HWSPWCH0 is called to process the special request.

To establish the HWSPWCH0 address, the HWSPWCH0 object code must be included in the exit routine and an **"INCLUDE TEXT(HWSPWCH0)"** statement added to the exit routine JCL for the binder (link-edit) step.

Value: You have dynamic control over password resets.

2.17.2 IMS Version 12: RACF return codes passed back to client

There is new support in the IMS Connect exits to pass any bad return code from RACF back to the client. This process is done by extending the header for the *REQSTS* RSM when it is returned to the client.

For more information about this facility, see *IBM IMS Version 12 Technical Overview*, SG24-7972.

Value: Clients have more control over their security status regarding IMS Connect return codes that are passed back to them.

2.17.3 IMS Version 12 / 13: RACF ENF support for cached RACF user IDs

IMS Connect Version 12 enables RACF UserID Caching by defining the parameter **UIDCACHE=Y** in the **HWS** statement in the HWSCFGx configuration or by using the following commands:

- ▶ Type-2 command **UPDATE IMSCON TYPE(CONFIG) SET(UIDCACHE(ON))**
- ▶ WTOR command **SETUIDC ON**
- ▶ z/OS command **UPDATE MEMBER TYPE(IMSCON) SET(UIDCACHE(ON))**

Example 2-35 presents the output of the **SETUIDC ON** WTOR command.

Example 2-35 Output of the SETUIDC ON command

```
R 184,SETUIDC ON
IEE600I REPLY TO 184 IS;SETUIDC ON
HWSP1501I RACF USERID CACHING ENABLED ,M=CHWS
```

Additionally, these cached user IDs can be refreshed based on an aging value or manually by issuing the following commands:

- ▶ **WTOR (xx,REFRESH RACFUID (NAME(userid))**
- ▶ **z/OS Modify (F hws,UPDATE RACFUID NAME(userid) OPTION(REFRESH))**
- ▶ **Type-2 command (UPDATE IMSCON TYPE (RACFUID))NAME(userid) OPTION(REFRESH))**

The IMS Version 13 enhancement automatically refreshes cached UIDs by listening to RACF events (the type 71 ENF signals that are produced by the following RACF commands: **CONNECT**, **REMOVE**, and **ALTUSER REVOKE**). IMS Connect then acts on the signal to refresh the affected UIDs.

This new capability is applicable only when RACF UID caching is enabled in IMS Connect.

Value: IMS Connect Version 12 enabled RACF UserID Caching for improved performance. Manual intervention is not needed to refresh those cached UIDs in IMS Version 13.

2.18 IMS Connect scalability and performance

IMS Connect provides scalability and performance.

2.18.1 IMS Connect performance parameters

Within the HWSCFG IMS Connect configuration member, there are parameters of performance interest:

- ▶ Set **ECB=Y** to post an ECB when there is work to do.
- ▶ **MAXSOC= xxxx** sets the maximum number of concurrent sessions for this instance of IMS Connect that can be opened. Specify a large enough value to support concurrent throughput requirements.
- ▶ The **NODELAY** parameter is used for output packets from the IMS Connect side. Specify **Y** to enhance IMS Connect performance and increase throughput because it forces a socket to send the data in its buffer without having to wait for an ACK from the client's Internet Protocol network.

2.18.2 TCP/IP performance parameters

TCP/IP is the backbone for communication for modern connectivity solutions. This section touches on a few important performance topics. You can choose different TCP/IP values to maximize your environment settings for IMS Connect.

The following **SETSOCKOPT** TCP/IP parameter values affect IMS Connect:

- ▶ **SO_KEEPAIVE**
The **SO_KEEPAIVE** parameter activates **KEEPAIVE** for this socket. The **SO_KEEPAIVE** option causes a packet (called a *keepalive probe*) to be sent to the remote system if a long time passes with no other data being sent or received. This packet provokes an ACK response from the peer, which enables detection of a peer that becomes unreachable (for example, powered off or disconnected from the net).
- ▶ **TCP_NODELAY=ENABLE**
Data is transmitted by TCP/IP per client **SEND**. TCP/IP waits one millisecond per transmission. Multiple client TCP/IP **SENDS** can result in multiple TCP/IP transmissions.

► **TCP_NODELAY=DISABLE**

Data is collected by TCP/IP from client TCP/IP SENDs before transmission. TCP/IP waits until the buffer is full before transmission. Multiple client SENDs results in 1 - n TCP/IP transmissions to IMS Connect.

► **SO_LINGER=Y, VALUE=0**

A client request to close the socket can bypass data that is sent with a previous client TCP/IP SEND request, but might result in the loss of the client SEND data.

► **SO_LINGER=N**

A client request to close the socket can bypass data that is sent with a previous client TCP/IP SEND request, but might result in the loss of the client SEND data.

► **SO_LINGER=Y, VALUE=10**

Return code to a client when an ACK is received from the host, or waits for 10 seconds before sending a close. A socket close does not bypass data that is sent.

Here are other parameters of interest to performance:

► **DELAYACK**

DELAYACK is used to minimize non-data transmissions from the host. If **DELAYACK** is used, TCP/IP waits 200 ms before sending an ACK to the remote server TCP/IP. However, if the ACK is appended to the data being sent from IMS Connect, there is no delay.

If your client application performs a single SEND followed by a READ, **DELAYACK** is recommended. **DELAYACK** can be set on the TCP/IP "Port Statement" or on the "Gateway Statement."

► **NODELAYACK**

NODELAYACK is used to allow non-data transmissions from the host to flow without data. If **NODELAYACK** is used, the z/OS TCP/IP immediately sends an ACK to the remote server TCP/IP. The ACK is not appended to the data being sent from IMS Connect.

If the client code sends one SEND followed by a READ to the host with a **NODELAYACK** setting, an ACK is sent separately.

If the client code sends two or more SENDs followed by a READ to the host, the host TCP/IP sends an ACK immediately to the data received, which allows the next SEND of data from the client to flow.

NODELAYACK is recommended if your client application sends more than one SEND followed by a READ.

2.19 IMS Connect scalability and performance enhancements

To support the ever increasing message volumes that go through it, IMS Connect is continuously enhanced to improve its ability to manage enterprise scope workloads.

2.19.1 IMS Version 10: The NODELAY parameter

The IMS Connect performance enhancement introduces the new **NODELAY** configuration parameter in the **TCPIP** statement. Using the **NODELAY** option can enhance IMS Connect performance and increase the throughput because it forces a socket to send the data in its buffer without having to wait for an ACK response from the client's TCP/IP. The **NODELAY** parameter is used only when sending packets.

Value: There is a performance enhancement when you send packets.

2.19.2 IMS Version 11: IMS Connect performance enhancement

A new hashing mechanism for client IDs enhances IMS Connect performance by reducing processor impact and increasing throughput. This support is also available in IMS Version 10 with APAR PK57574 through PTF UK42318.

Value: There is a performance improvement for IMS Connect message processing, which is obtained by enhancing the client scan process.

2.19.3 IMS Version 12: RACF user ID caching

User ID caching is a new function in IMS Version 12 to reduce IMS Connect storage usage and reduce the number of calls that are made to RACF to check user IDs. If the same user ID connects from more than one client or on more than one port, IMS Connect searches its cache to find security credentials (assuming they have not expired because of aging). If the user ID is not found, only then does it call RACF.

Example 2-36 shows the new specification for **UIDCACHE** (whether IMS Connect caches verified RACF user IDs for reuse) and **UIDAGE** (specifies the refresh interval for cached RACF user IDs in seconds). **UIDAGE** provides better control when IMS Connect removes credentials from storage and calls RACF to renew them. **UIDCACHE** can be controlled by an external command. **UIDAGE** can be controlled only by updating the HWSCFG member and restarting IMS Connect.

Example 2-36 Sample IMS Connect configuration member with the RACF user ID caching specification

```
HWS=(ID=HWSI12A1,  
PSWDMC=R,RACF=Y,  
UIDCACHE=Y,UIDAGE=500,  
RRS=Y,XIBAREA=50)
```

Three new commands control user ID caching. Table 2-7 shows the commands to control user ID caching and to refresh the in-storage data for a user ID.

Table 2-7 Commands for User ID caching

Type-2 command	WTOR reply	Modify command
UPD IMSCON TYPE(CONFIG) SET(UIDCACHE(ON))	SETUIDC ON	UPDATE MEMBER TYPE(IMSCON) SET(UIDCACHE(ON))
UPD IMSCON TYPE(CONFIG) SET(UIDCACHE(OFF))	SETUIDC OFF	UPDATE MEMBER TYPE(IMSCON) SET(UIDCACHE(OFF))
UPDATE IMSCON TYPE(RACFUID) NAME(userid) OPTION(REFRESH)	REFRESH RACFUID NAME(userid)	UPDATE RACFUID NAME(userid) OPTION(REFRESH)

Value: There is Improved scalability and performance by reducing IMS Connect storage usage and reducing the number of calls that are made to RACF to check user IDs.

2.19.4 IMS Version 13: Usage of the internal CPOOL storage macro

Before IMS Connect Version 13, the STORAGE OBTAIN and STORAGE RELEASE macros were used to obtain virtual storage. This enhancement uses the CPOOL macro to obtain virtual storage in modules that make frequent calls for storage obtain and release. The usage of the CPOOL macro is a more efficient method for obtaining and releasing virtual storage.

Value: Internal IMS Connect efficiencies are realized.

2.20 IMS Connect diagnostic tests

There are several facilities that are available to you to diagnose IMS Connect situations.

- ▶ The IMS Connect recorder trace is a function to capture information that is related to IMS Connect input and output data to a data set. The IMS Connect recorder trace is also called the IMS Connect line trace.
- ▶ Through the IMS Connect Base Primitive Environment, IMS Connect enables you to trace diagnostic information about events going on within the address space.
- ▶ The IMS Connect Dump Formatter can be used to format IMS Connect internal control blocks under the control of the Interactive Problem Control System (IPCS).

2.21 IMS Connect diagnostic enhancements

There have been advancements in IMS Connect diagnostic tests since their introduction in IMS Version 9.

2.21.1 IMS Version 10 and 11: HWSTECL0 event record enhancements

IMS Connect facilitates event recording by passing event data to the load module, HWSTECL0. This module stores all trace and XML event notifications through a recording routine and can be used by any event recording function. IMS Version 11 enhances the capabilities of HWSTECL0 by adding an event for IMS Connect Event #45, based on the OTMA protocol message, and identifies whether any of the OTMA resources are in a severe, warning, or normal state.

Value: There is improved event tracing of OTMA activities.

2.21.2 IMS Version 11: Usage of the BPE External Trace facility

Extensions of the Base Primitive Environment (BPE) External Trace facility for the recorder trace were introduced. The extended BPE Direct External Trace enables BPE to write ad hoc data of variable length directly to a copy buffer. The recorder trace uses the BPE Direct External Trace facility by rerouting recorder trace data to the BPE External Trace data set.

Value: There are improved diagnostic tests through the usage of a larger BPE External Trace data set.

2.21.3 IMS Version 11: Additional information in message HWSP1410W

To help diagnose problems that might occur when IMS Connect frees storage that is used for buffers, IMS Version 11 adds the address of the buffer that is associated with the error to the existing message HWSP1410W. This message is associated with failures to release storage for internal buffers.

Value: There is improved problem source identification.

2.21.4 IMS Version 11: Warning messages and early detection of maximum sockets

IMS Connect Version 11 supports 50 - 65,535 sockets. The maximum number of sockets that an IMS Connect instance supports is specified in the **MAXSOC=** parameter. When the number of sockets reaches the **MAXSOC** limit, any new connections are refused and message HWSS0771W is issued. IMS Version 11 introduced the **WARNSOC** and **WARNINC** parameters as ways to provide early detection of this potential problem. **WARNSOC** supports a decimal value of 50% - 99% as a warning level.

Value: There are improved early warning facilities that are associated with socket usage

2.21.5 IMS Version 12: IMS Connect Recorder Trace

IMS Version 12 adds a level of tracing to IMS Connect. It adds records for TCP/IP and MSC, and the cross-system coupling facility (XCF) sends and receives at the trace points that are listed in Table 2-8.

Table 2-8 IMS Connect additional recorder trace entries

Trace point eye catcher	What is traced	Tracing method
ICONTR	TCP/IP Receive	BPE
ICONTS	TCP/IP Send	BPE
ICONIR	IMS OTMA Receive	BPE
ICONIS	IMS OTMA Send	BPE
ICONMS	MSC send	Old tracing
ICONMR	MSC receive	Old tracing
ICONRR	Remote IMS Connect to local IMS Connect	Old tracing

Value: There is improved tracking of IMS Connect information flows.

2.21.6 IMS Version 13: Expanded Recorder trace records

As described in 2.21.5, “IMS Version 12: IMS Connect Recorder Trace” on page 82, IMS Connect Version 12 introduced new Recorder Trace Records that use BPE external trace tables. In IMS Connect Version 13, these trace records are expanded to capture the entire messages that are sent and received on the DRDA sockets, and all messages that are sent and received to and from OM and ODBM through SCI.

Changing the BPE trace level from MEDIUM to HIGH is the trigger to write the enhanced trace records. Additionally, they must be written to the external trace data set because they are not written to the fixed size HWSRCDR trace data set.

The macro HWSUSTAT containing the mapping of the recorder trace records was enhanced and converted to non-Object Code Only (non-OCO) format. The macro HWSUSTAT is shipped as a member of SDFS MAC.

Value: There are additional levels of tracing that are available for diagnostic and performance purposes.



IMS Connect Extensions for z/OS

IBM IMS Connect Extensions for z/OS (IMS Connect Extensions) (product number 5655-S56) enhances the manageability of TCP/IP access to IMS through IMS Connect.

This chapter provides a general overview of IMS Connect Extensions and examples of problem diagnosis, operational management, and open database access by using the product.

This chapter covers the following topics:

- ▶ Overview
- ▶ Scenarios
- ▶ Scheduled and unscheduled outages
- ▶ Application development considerations

3.1 Overview

IMS Connect Extensions provides operational management and controls for IMS Connect. Specifically, the tool provides the following functions:

- ▶ **Monitoring and recording of TCP/IP activity**
This function records and journals TCP/IP activity through IMS Connect, including transactional (OTMA) and database (ODBM) requests. The information helps you analyze performance, throughput, resource availability, and security. You can also use this information to debug clients and new applications.
- ▶ **Message routing and shaping**
This function dynamically shapes message activity based on availability, capacity, flood-state, and more to provide greater resilience, improved parallelism, simplified system management, and better cost controls.
- ▶ **Operational management**
The ISPF, a graphical user interface (GUI), and batch-based management of IMS Connect. This includes real-time activity monitoring, basic controls, OPERLOG¹ triggerable events, and analytical tools (filtering, highlighting, summarizing, and exporting). Improves the management of clients by centrally managing client options.
- ▶ **Enhanced security**
ACEE caching, system-based authentication, and IP address validation. Includes enhanced auditing of security events and the ability to reject messages matching certain criteria.

These features enable you to perform the following functions:

- ▶ Improve the availability, reliability, and performance of IMS Connect
- ▶ Speed and simplify problem determination
- ▶ Make your systems more transparent so that they are easier to audit and manage

IMS Connect Extensions consists of components that run with IMS Connect, journal data sets that record IMS Connect activity, and an ISPF dialog-based client and Operations Console GUI client for managing IMS Connect systems and their IMS Connect Extensions features.

We examine the following events:

- ▶ Event collection
- ▶ Routing
- ▶ Operational controls

3.1.1 Event collection

IMS Connect Extensions provides a comprehensive picture of activity within IMS Connect by recording information directly from IMS Connect. As Figure 3-1 on page 87 shows, information about activity in IMS Connect is written to journals while the activity that occurs within OTMA and IMS is recorded in the IMS logs. Other IBM IMS Tools can combine the data to provide an end-to-end picture of activity for a transaction.

¹ The MVS operations log (OPERLOG) function of the MVS console services component provides a sysplex-wide merged message log by using the MVS system logger services.

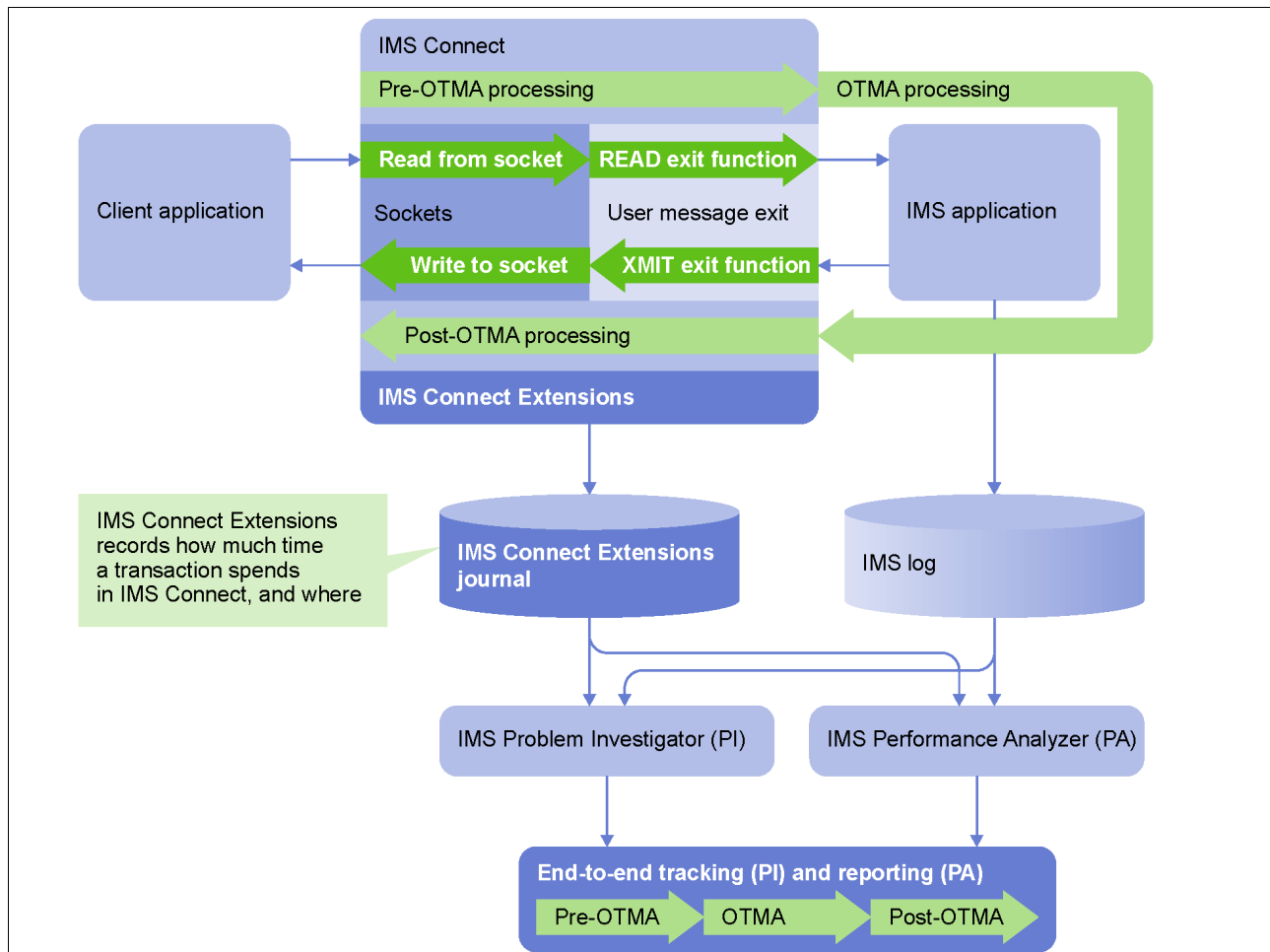


Figure 3-1 IMS Connect Extensions journals combine with the IMS log to offer information about TCP/IP transactions

IMS Connect Extensions also provides event collection for IMS Open Database, which allows application developers to use the open DRDA protocol and SQL to access IMS data through IMS Connect.

In the context of Open Database, we can divide the information that IMS Connect Extensions collects into these categories:

- ▶ Framing events
- ▶ Application-level events

Framing events are events that mark significant points in the lifecycle of an Open Database request. Each such event contains a time stamp that gives you information about the relative timings and duration of the entire DRDA request. These events include the following ones:

- ▶ Opening of the TCP/IP socket
- ▶ Processing of the DRDA request
- ▶ Security authentication and authorization
- ▶ Dispatch of requests to ODBM address space
- ▶ Response that is received from ODBM address space
- ▶ Processing of response
- ▶ Dispatching to client
- ▶ Syncpoint processing (if RRS is used)

Such events help you understand the flow of an Open Database request and obtain timings not only for the overall request but for individual processing steps for the request.

Conversely, application-level events provide a record of exactly what request was sent by the client and what response was received back from IMS. They provide a complete breakdown of individual DDM objects and data. They help you debug and tune applications and audit and monitor exactly what activity is being performed by clients.

As shown in Figure 3-2, like traditional transactional events, open database events are also collected in the IMS Connect Extensions journals and can be combined with information that is collected in the IMS log for end-to-end analysis of open database requests.

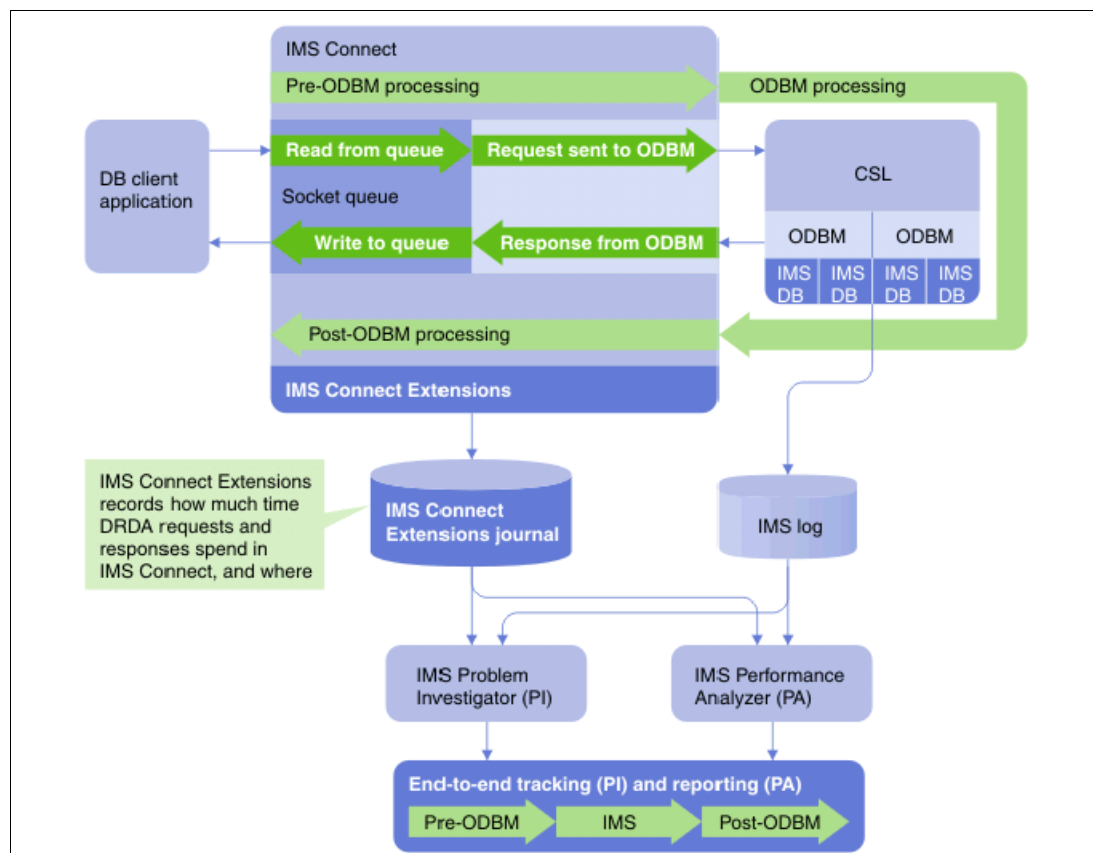


Figure 3-2 IMS Connect Extensions also records Open Database requests

Analysis in IMS Problem Investigator

Using the IBM IMS Problem Investigator for z/OS tool, you can analyze your IMS environment by interrogating IMS, CQS, monitoring logs, DB2 logs, and SMF data information that is collected by IBM OMEGAMON® ATF and TRF, and reviewing the information that is collected in IMS Connect Extensions journals. This section focus on the analysis of the IMS Connect Extensions journals.

As shown in Figure 3-3 on page 89, by allowing you to navigate, format, query, and extract these journals, you can better understand and gain insight into a TCP/IP request. IMS Problem Investigator can identify records that are related to a request sequence through IMS Connect and track related log records right through IMS and other related subsystems.

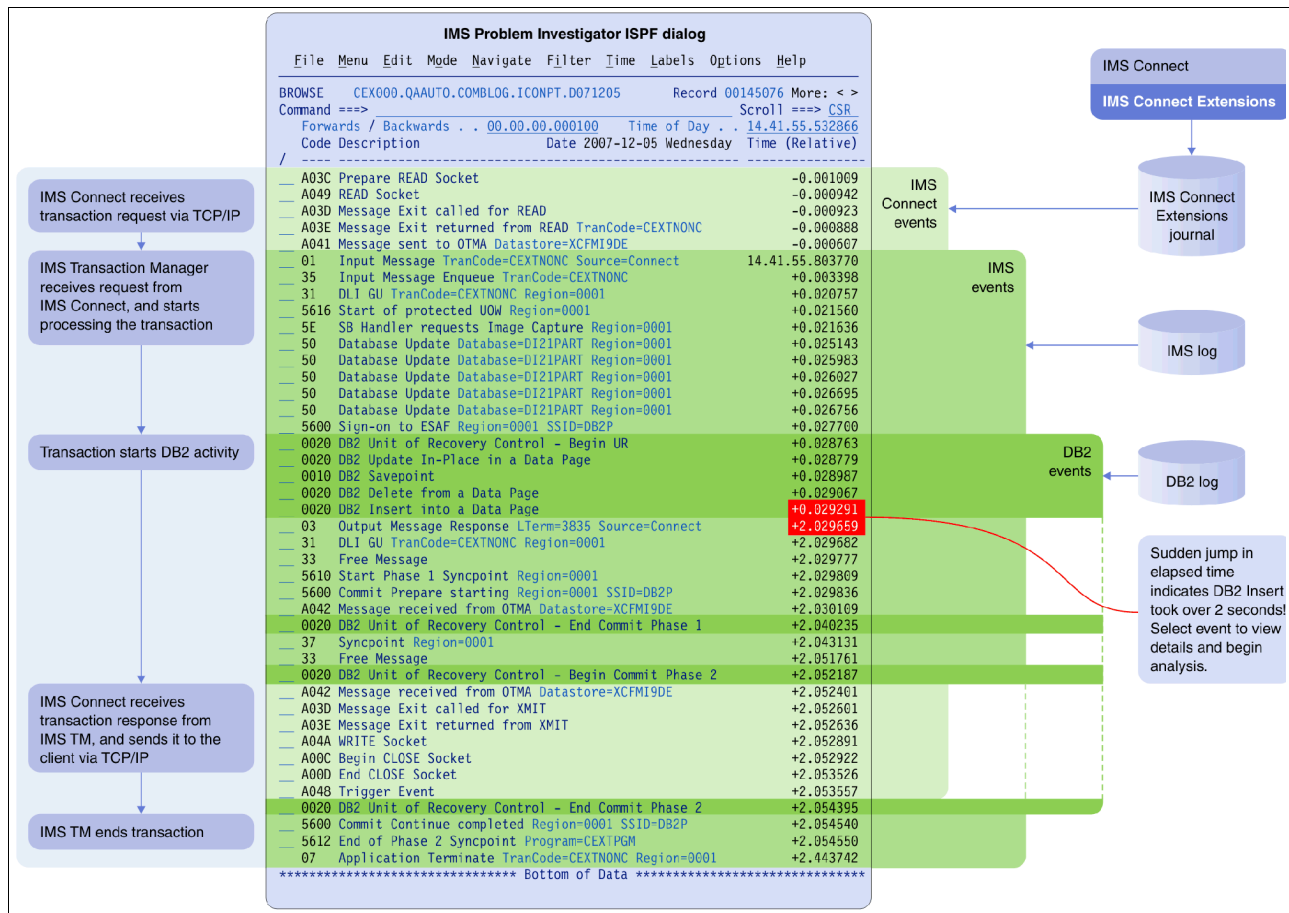


Figure 3-3 IMS Problem Investigator tracks all the records for a transaction from across IMS Connect, IMS, and DB2

This process can be further refined by using the IMS Connect Transaction Index that IMS Performance Analyzer produces from IMS Connect Extensions journals. As shown in Example 3-1, the index contains a consolidated record of the transaction characteristics.

Example 3-1 IMS Problem Investigator display of key fields in an index record

CA20 Connect Transaction	02.24.53.281040
TranCode=PRT2 Userid=CEX2 IMSID=ICDH ClientID=CLNAC02 Port=3801	
LogToken=C8962E0E2AF33103 SSN=0689 Response=0.041073 CM=1 SYNCLEVEL=1	
TOV=38_MIN Socket=Tran	

The index allows you to search in IMS Problem Investigator for transactions that match certain overall characteristics and then track all the related records. For example, search for records where the response time was greater than two seconds. For more information, see "Reporting in IMS Performance Analyzer" on page 89.

Reporting in IMS Performance Analyzer

IMS Connect Extensions allows you to incorporate IMS Connect performance metrics as part of your reporting. This means you can obtain the following information:

- ▶ Transaction response times, including IMS Connect responses
- ▶ A breakdown of activity within IMS Connect

This information allows you to identify performance issues with IMS Connect and to eliminate IMS and IMS Connect as a source of performance problems. As shown in Figure 3-4, without the reporting from IMS Connect, there is a performance “black hole” that makes it difficult to assign and attribute problems to various subsystems.

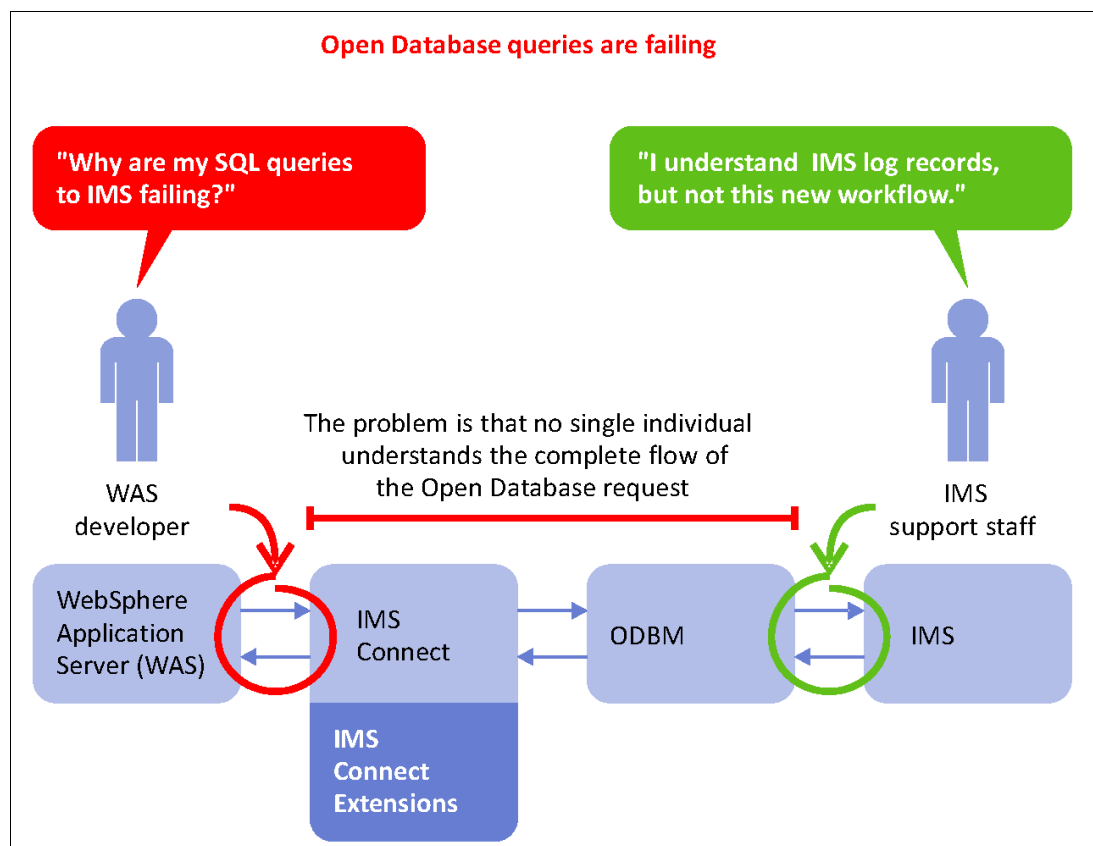


Figure 3-4 Without IMS Connect event recording there is a performance gap that can stifle cooperation between groups

For example, Figure 3-5 on page 91 shows a scenario where IMS reports millisecond responses but the client experiences a long (greater than 1 second) response time. Without IMS Connect reporting, it is not possible to account for this gap, but using the information that is collected by IMS Connect Extensions, it is possible to account for this gap. The report shows that most of the delay is accounted for by time that is spent before the IMS transaction is placed on the input queue.

IMS Performance Analyzer									
Combined tran list									
LIST0001 Printed at 19:33:38 12Dec2007					Data from 13.57.52 12Dec2007				
CON Tran			CON Resp	PreOTMA	OTMAproc	IMS Tran	InputQ		
Start	Trancode	OTMA	Time	Time	Time	Start	Time		
13.57.52.714	IMSTRANS	CONNECT	1.810	0.000	1.803	13.57.54.517	0.000		
13.57.52.964	IMSTRANS	CONNECT	1.575	0.000	1.574	13.57.54.538	0.000		
13.57.52.972	IMSTRANS	CONNECT	1.588	0.000	1.588	13.57.54.548	0.009		
13.57.53.091	IMSTRANS	CONNECT	1.716	0.002	1.714	13.57.54.806	0.000		
13.57.53.567	IMSTRANS	CONNECT	1.839	0.000	1.839	13.57.55.403	0.000		
13.57.54.044	IMSTRANS	CONNECT	1.800	0.000	1.799	13.57.55.836	0.006		

Process	Total	PostOTMA
Time	IMS	Time
0.001	0.001	0.006
0.001	0.001	0.000
0.002	0.011	0.000
0.001	0.001	0.000
0.002	0.002	0.000
0.001	0.007	0.001

Figure 3-5 IMS Connect Extensions is required to gain an end-to-end report of TCP/IP transaction performance

If you look at the second part of the report, you can discover that the delay is occurring in OTMA rather than IMS Connect by seeing that time before (PreOTMA) and after (PostOTMA) OTMA are also subsecond. Although statistics such as input queue time and IMS processing time are available from the IMS log, IMS Connect Extensions is required for reporting the remaining statistics. For a more detailed explanation of the call-flow that is shown in this report, see Figure 3-6 on page 92.

Transit analysis reports

IMS Performance Analyzer Version 4.2, program number 5655-R03, is a component of IMS Performance Solution Pack, and provides more in-depth analysis and reporting of IMS Connect event records.

IMS Performance Analyzer provides a comprehensive set of reports from the IMS Connect performance and accounting data that is collected by IMS Connect Extensions. The reports provide a summary and detailed analysis of IMS Connect transaction transit time, resource usage, and resource availability.

The IMS Connect Transit Analysis report provides a summary of IMS Connect transaction performance. Performance data can be summarized by one or two sort keys, including time of day, transaction code, user ID, data store (original and target), and port number.

Performance statistics are provided as averages, and optionally, peak percentiles. For example, you can specify 90 to report the elapsed time within which 90% of the transactions completed. To be complete, this report requires IMS Connect Extensions to collect event data at collection level 3 or 4.

Example 3-2 shows a Transit Analysis report for sync level none transactions with no RACF security activated in IMS Connect. The transaction peak is set to 80%, and the data is summarized by transaction code.

Example 3-2 Transit Analysis report for sync level none transactions

IMS Performance Analyzer 3.3													
IMS Connect Transit Analysis - IMSGCONN													
From 13Jun2005 18.16.15.36 To 20Jun2005 14.47.43.06													
Transact Code	Message Count	Response Time	Input		-Process-		Output		Rate Time		Page 1		
			Pre-OTMA	READ Sock	READ Ex	SAF	OTMA	Confirm	Post-OTMA	XMIT Ex	/Sec	Outs	NAK
IVTNO	234	Avg 9.284	0.229	0.052	0.025	0.000	8.794	0.000	0.261	0.027	0	0	0
		80% 23.329	0.396	0.077	0.033	0.000	22.826	0.000	0.371	0.040			
PART	50	Avg 103.958	0.332	0.057	0.129	0.000	37.566	0.000	66.058	0.041	0	0	0
		80% 154.813	0.934	0.086	0.655	0.000	86.206	0.000	96.568	0.068			
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----
Total	284	Avg 25.952	0.247	0.053	0.043	0.000	13.859	0.000	11.845	0.029	0	0	0
		80% 65.134	0.541	0.078	0.265	0.000	39.495	0.000	36.503	0.046			

The report gives the total response time and the intermediate times that are divided by groups: input, process, and output times. It helps you to identify the cause of bad response times.

The IMS Performance Analyzer uses the connect events record time stamps to give the average values of the IMS Connect transaction performance. Figure 3-6 uses the event flow for sync level none transactions to show graphically the meaning of the most relevant fields of the Transit Analysis report.

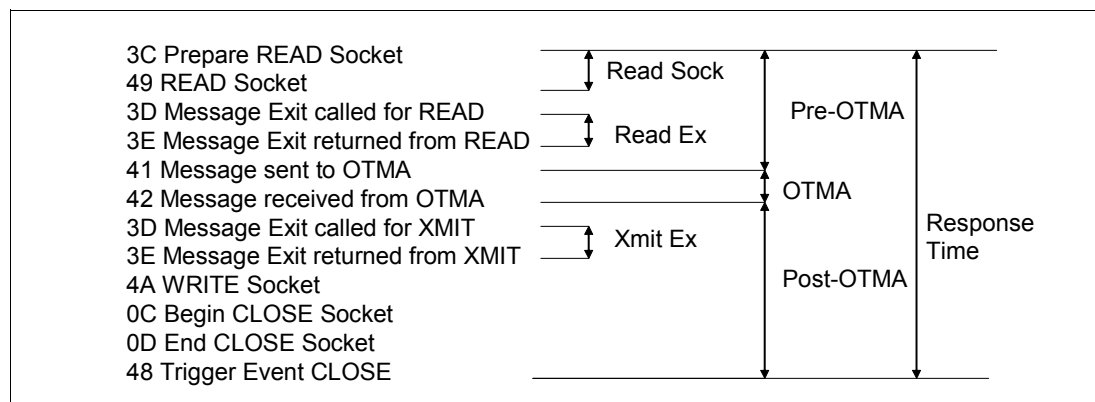


Figure 3-6 Transit Analysis report fields for sync level none transactions

The Transit Analysis report also gives timeout and NAK counters. In Example 3-3, you can see the Transit Analysis report in the same environment as in Example 3-2, but with sync level confirm transactions.

Example 3-3 Transit Analysis report for sync level confirm transactions

IMS Performance Analyzer 3.3															
IMS Connect Transit Analysis - IMSGCONN															
Transact Code	Message Count		From 13Jun2005 19.23.57.91 To 20Jun2005 16.36.47.51								Page 1		Rate /Sec	Time Outs	NAK
			Response Time	Pre-OTMA	Input READ Sock	READ Ex	SAF	-Process- OTMA	Confirm	Output Post-OTMA	XMIT Ex				
IVTNO	199	Avg	423.346	0.244	0.050	0.059	0.000	6.430	91.360	215.038	0.056	0	199	0	
		80%	509.536	0.425	0.071	0.083	0.000	16.250	118.189	280.791	0.093				
PART	42	Avg	223.626	0.210	0.048	0.094	0.000	32.199	111.543	79.672	0.055	0	0	0	
		80%	246.098	0.276	0.060	0.140	0.000	41.315	127.335	80.381	0.060				
-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	
Total	241	Avg	388.540	0.238	0.049	0.065	0.000	10.921	94.877	191.447	0.056	0	199	0	
		80%	490.035	0.405	0.069	0.096	0.000	23.640	120.919	265.232	0.090				

The report changes for the sync level confirm transaction. A new field, Confirm, appears, and fields such as OTMA and the fields that are related to exits now have a different meaning.

Figure 3-7 uses the event flow for sync level confirm transactions to show graphically the meaning of the most relevant fields of the Transit Analysis report. Now, the OTMA and exits average time is calculated as the addition of two different times.

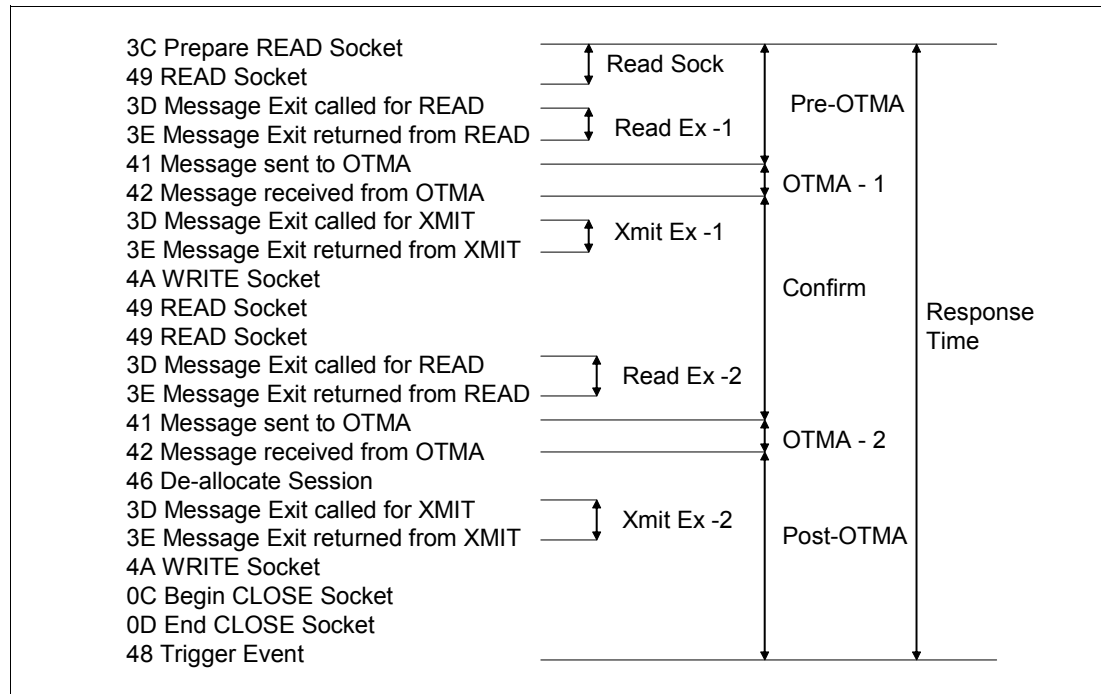


Figure 3-7 Transit Analysis report fields for sync level confirm transactions

To focus on the meanings of the time fields, the previous examples show the total average times for every transaction, but you are able to obtain the same results by time intervals with total and subtotal times.

Example 3-4 shows this Transit Analysis report option using one-minute time intervals.

Example 3-4 Transit Analysis report by transaction and one-minute time with subtotals

IMS Performance Analyzer 3.3 IMS Connect Transit Analysis - IMSGCONN															
From 20Jun2005 17.17.22.77 To 20Jun2005 17.22.09.84															
Transact Code	Time	Message Count		Response Time	Input				-Process- OTMA	Confirm	Output		Page Rate /Sec	Time Outs	NAK
					Pre-OTMA	READ Sock	READ Ex	SAF			Post-OTMA	XMIT Ex			
DSPINV	17.20.00	1	Avg	222.965	0.220	0.043	0.075	0.000	41.166	101.635	79.943	0.070	0	0	0
			80%	222.965	0.220	0.043	0.075	0.000	41.166	101.635	79.943	0.070			
	17.21.00	4	Avg	213.761	0.244	0.070	0.171	0.000	25.613	101.011	86.892	0.052	0	0	0
			80%	229.149	0.294	0.087	0.321	0.000	26.738	112.305	94.814	0.058			
	17.22.00	1	Avg	439.132	0.189	0.040	0.084	0.000	38.251	105.004	295.687	0.054	0	0	0
			80%	439.132	0.189	0.040	0.084	0.000	38.251	105.004	295.687	0.054			
DSPINV	Subtotal	6	Avg	252.857	0.231	0.061	0.141	0.000	30.312	101.781	120.533	0.055	0	0	0
			80%	330.675	0.274	0.079	0.263	0.000	36.550	110.632	193.081	0.063			
PART	17.17.00	8	Avg	232.005	0.239	0.044	0.112	0.000	27.809	117.323	86.633	0.055	0	0	0
			80%	247.871	0.314	0.048	0.178	0.000	30.585	132.627	94.044	0.060			
	17.18.00	3	Avg	265.415	0.243	0.083	0.079	0.000	68.941	106.098	90.131	0.094	0	0	0
			80%	317.124	0.263	0.113	0.082	0.000	127.488	118.752	101.920	0.124			
	17.21.00	4	Avg	198.215	0.198	0.072	0.066	0.000	28.143	84.567	85.306	0.051	0	0	0
			80%	245.768	0.228	0.103	0.083	0.000	32.060	117.586	92.001	0.065			
PART	17.22.00	1	Avg	229.447	0.178	0.041	0.079	0.000	25.839	111.343	92.085	0.053	0	0	0
			80%	229.447	0.178	0.041	0.079	0.000	25.839	111.343	92.085	0.053			
	Subtotal	16	Avg	229.662	0.225	0.058	0.092	0.000	35.482	106.655	87.298	0.061	0	0	0
			80%	266.838	0.282	0.081	0.141	0.000	61.158	133.529	94.799	0.080			

Total	22	Avg	235.988	0.227	0.059	0.105	0.000	34.072	105.326	96.362	0.060	0	0	0
		80%	286.070	0.279	0.080	0.180	0.000	56.074	128.521	134.520	0.076			

3.1.2 Routing

The purpose of routing is to set a target data store (OTMA) or alias (Open Database) that is most appropriate based on dynamic criteria, such as the capacity of the targets, their state (available/stressed/unavailable), and other criteria in the message itself.

You can modify IMS Connect message exits to build routing or you can use tools that provide this capability. This section looks at IMS Connect Extensions as one such tool. The advantage of using tools is that it reduces the need for programmatic customization of IMS Connect and providing a flexible mechanism for ongoing changes.

Routing is valuable for various reasons:

- ▶ Improving availability

By providing multiple candidate data stores or aliases for a target that is specified by the message, routing provides multiple pathways for message processing. Routing tracks which candidates are unavailable or in stress and makes them ineligible as targets.
- ▶ Improving performance

By increasing parallelism through OTMA, routing helps improve transactional performance in some cases. Routing also allows you to use only local resources when they are available, and remote resources when the local resources are unavailable.
- ▶ Reducing costs

By routing messages to targets based on cost-centric requirements, you use lower-cost processing when it is available, but use only high-cost processing capabilities when it is essential.
- ▶ Protecting client applications from change

By allowing the client application to specify an *abstracted* target for processing and then letting routing resolve this target to a *concrete* target based on criteria. For example, the client might specify “BANKING” as the target for processing certain types of transactions, and this can be resolved to a changing set of physical data stores (DS01, DS02, DS03, and so on).

Configuring routing for availability and performance

Increased parallelism through OTMA helps improve transactional performance in some cases. Using local resources when they are available and using remote resources only when they are required can also help improve performance.

Routing provides multiple candidate targets for a candidate that is specified by the message. Routing tracks which candidates are unavailable or in stress and makes them ineligible as targets.

As shown in Figure 3-8, configuring routing in IMS Connect Extensions requires the following conditions:

- ▶ Defining conditions for activating the rule: Does the original data store on the message match the condition of the rule?
- ▶ Defining the message types for the rules: Routing is possible for transactional, send only, and synchronous, and a-synchronous calling.
- ▶ Defining target candidates and fallback candidates: Both are lists of data stores. The fallback list is used only if all the data stores in the target list are unavailable or under stress.

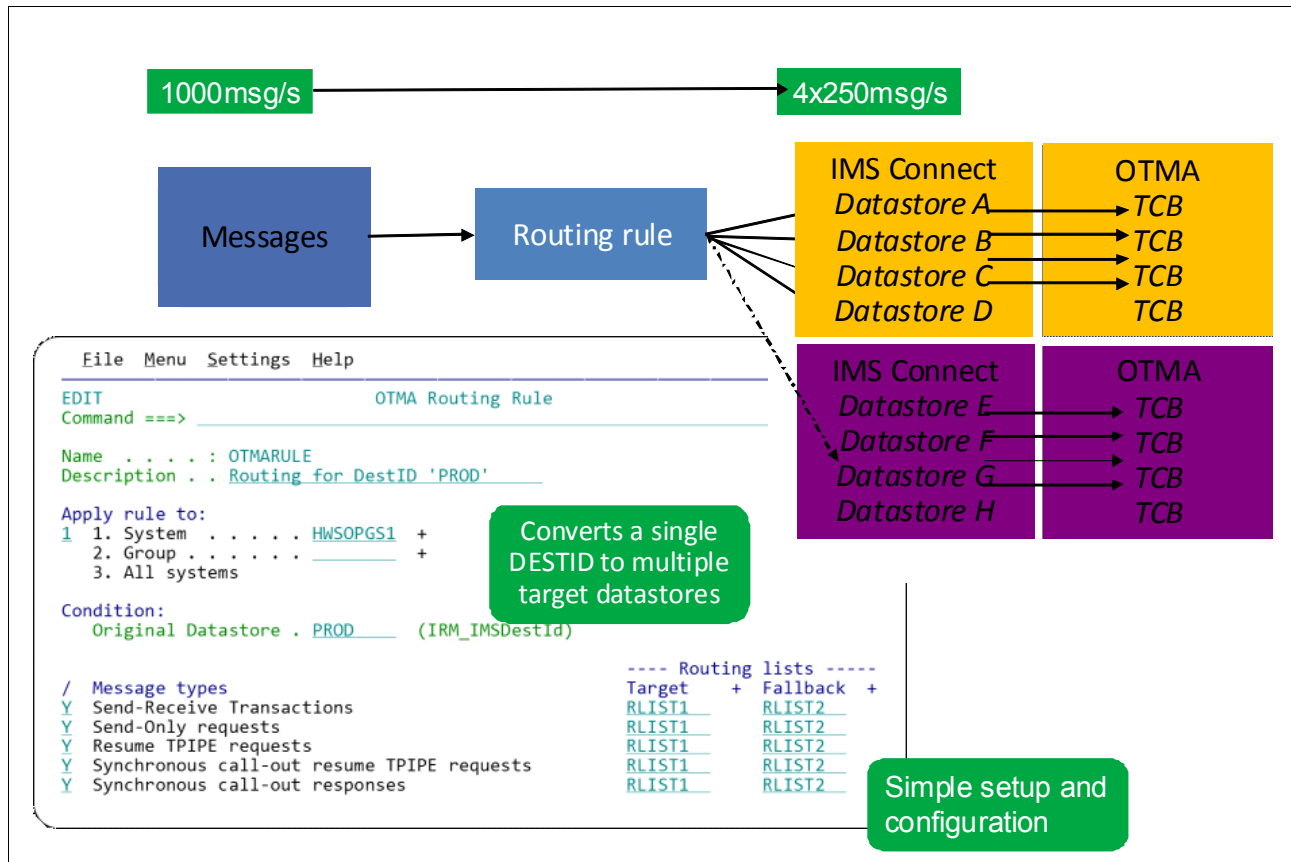


Figure 3-8 Routing configuration and the corresponding result

Configuring routing for traffic shaping

Routing allows you to set capacity weight ratings for individual data stores and Open Database alias targets. Capacity weights allow you to shape traffic volumes by setting a higher probability that a certain target is chosen for routing.

As shown in Figure 3-9, by changing capacity weights, you can offload capacity to cheaper processing (shown in red) when it can handle the throughput and free up higher-cost resources for other usages.

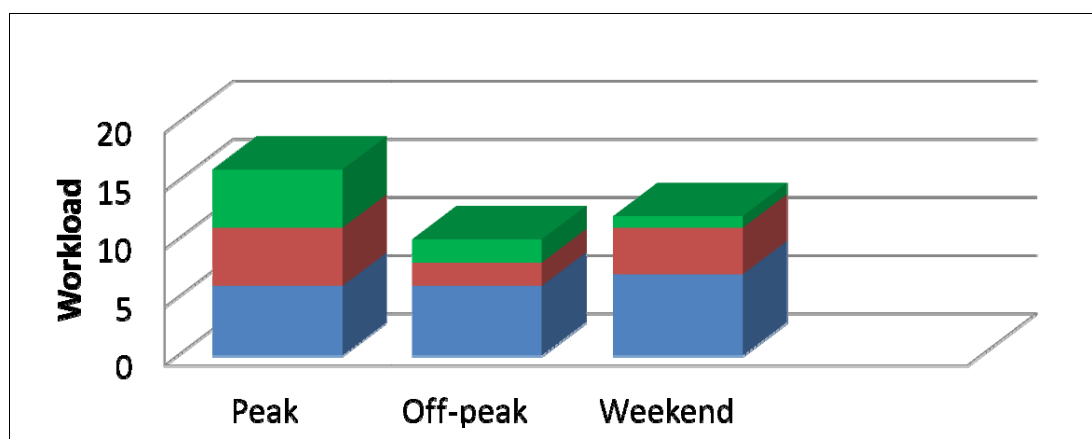


Figure 3-9 Shaping routing for different network demand

To use capacity weights, modify the definition of the target resources in IMS Connect Extensions and then refresh the definitions to implement the changes dynamically.

3.1.3 Operational controls

IMS Connect Extensions offers operational controls in batch, ISPF, and through the operations console, an Eclipse-based GUI. The operational controls allow you to perform the following actions:

- ▶ View real-time statistics about TCP/IP activity.
- ▶ View and control active sessions.
- ▶ View and control IMS Connect ports, data stores, and aliases.
- ▶ Refresh definitions.
- ▶ Reload and add message exits.
- ▶ Activate and deactivate traces.
- ▶ Submit IMS Connect WTOR commands and IMS Connect z/OS commands (IMS Connect OM commands are available through the GUI if you have IMS Configuration Manager).
- ▶ Submit IMS type I commands.

As shown in Figure 3-10 on page 97, one of the key objectives of the operational controls is to allow you to manage collections of systems in a single interface. For example, if you want to identify an active TCP/IP session, you do not need to know which IMS Connect is processing the message.

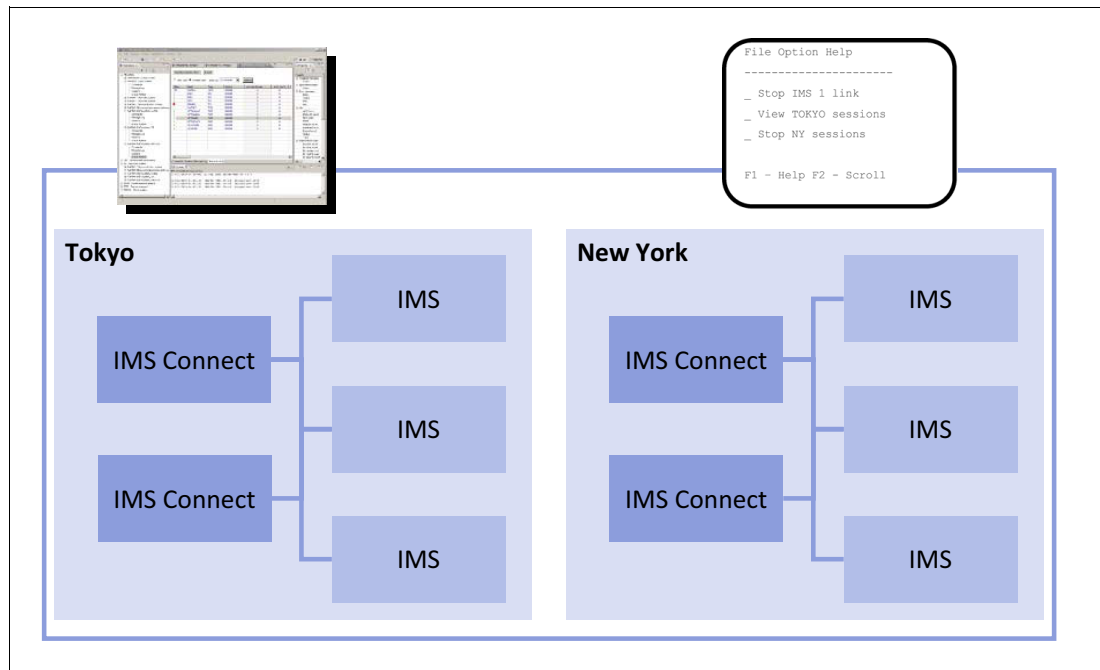


Figure 3-10 Centrally manage IMS Connect instances in ISPF or using the GUI

3.2 Scenarios

This section outlines common TCP/IP problems and resolutions, including the following events:

- ▶ Incorrect message length
- ▶ Identifying IMS problems with TCP/IP information
- ▶ Client fails to ACK
- ▶ OTMA timeouts
- ▶ Duplicate clients
- ▶ Open database issues
- ▶ Synchronous callout issues
- ▶ OTMA flood condition

3.2.1 Incorrect message length

IMS Connect does not process an input message until the entire the message is read. The initial LLLL value of the message controls the total length of the message. If the message contains an erroneous LLLL value, you can find problems in IMS Connect. For example, if the client specifies a larger LLLL value, the connection hangs because IMS Connect is expecting more data.

Identifying this problem is difficult because IMS Connect exits have not begun processing the message and because the problem is related to a hanging session, key information about it is still *active* on the IMS Connect system.

You can use the active session utility (in ISPF, batch, or the operations console GUI) to look at the active sessions and identify sessions with missing client IDs that are in the state of "reading remote client input".

In the example in Figure 3-11, we use the IMS Connect Extensions GUI to search for active sessions matching those characteristics.

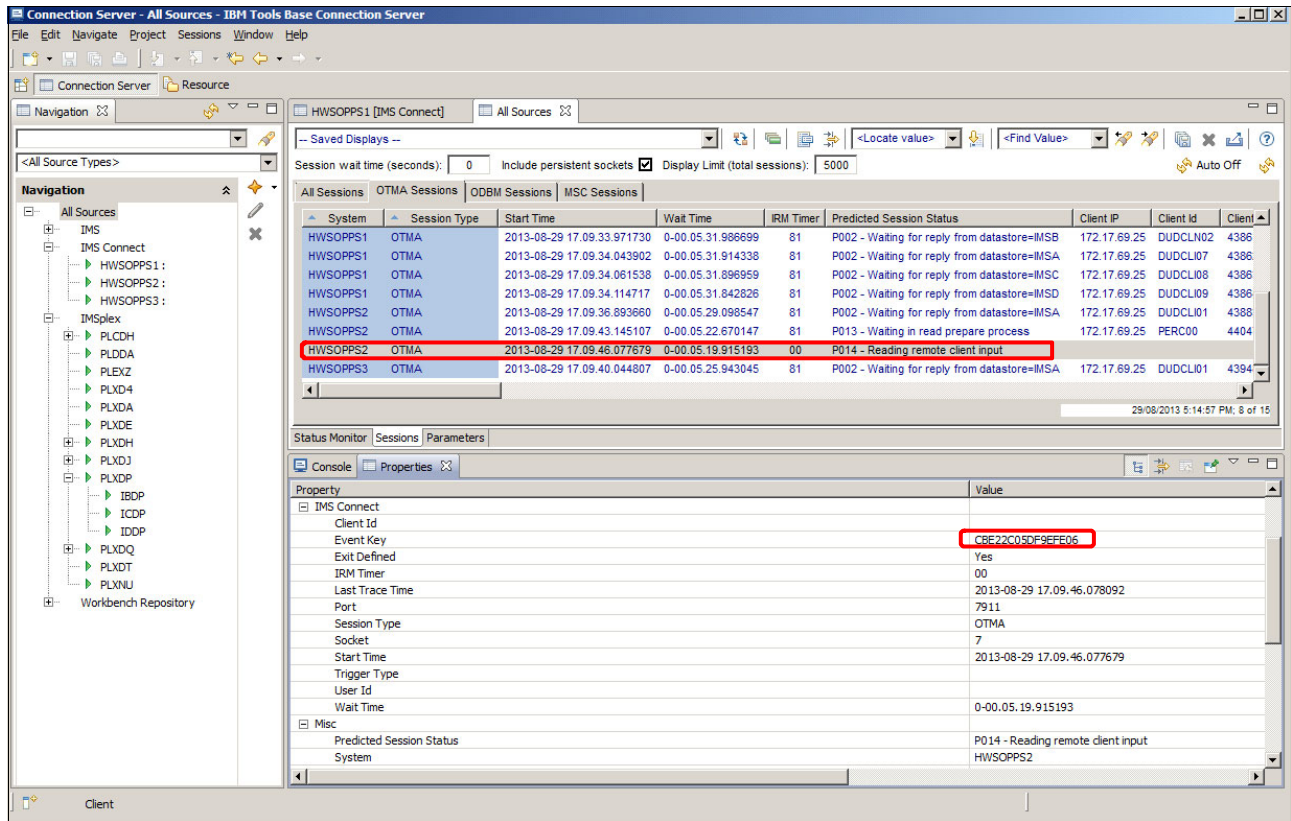


Figure 3-11 Obtaining the event key for a hanging session in the GUI

If you look at the IMS Connect Extensions journals in IMS Problem Investigator, you can use the Event Key to find the records and then select the record to see the request length and actual length that was read, as shown in Figure 3-12.

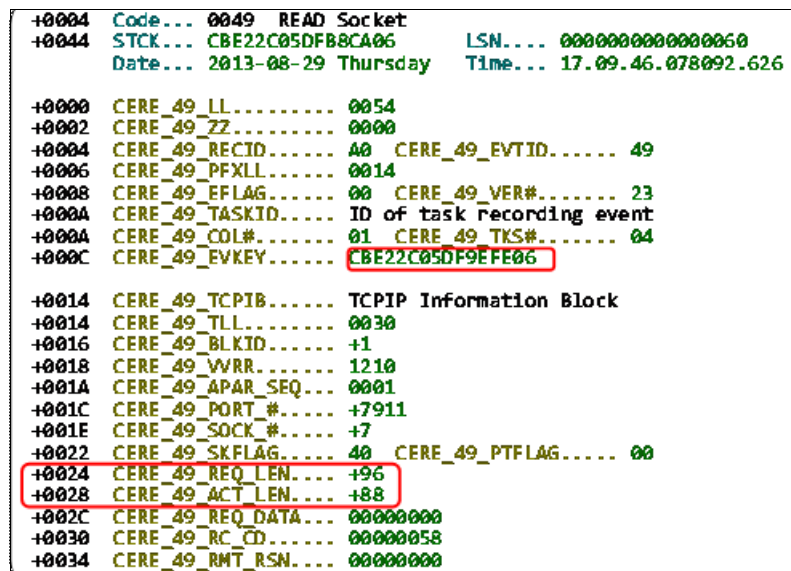


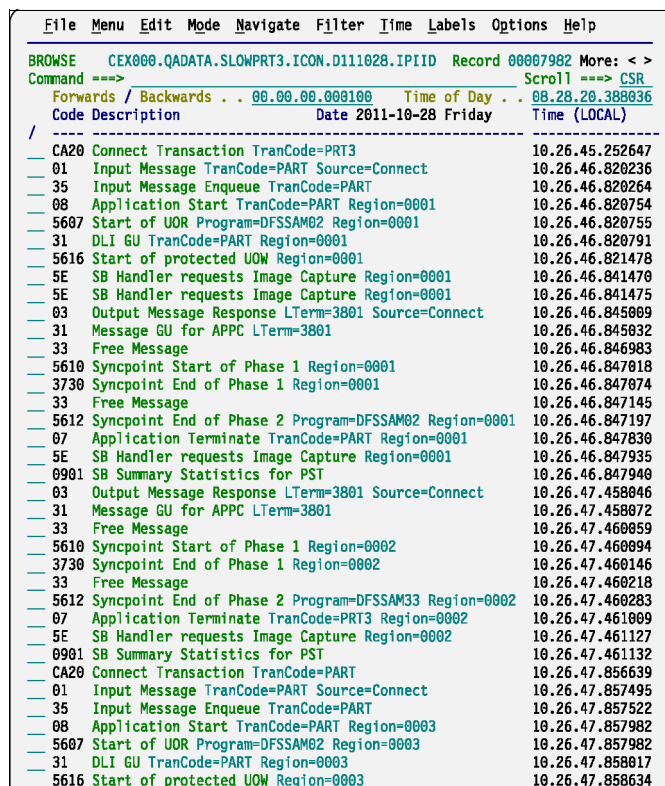
Figure 3-12 The difference between the request length and actual length indicates an incorrect message length

3.2.2 Identifying IMS problems with TCP/IP information

The IMS Connect Transaction Index is generated by IMS Performance Analyzer from IMS Connect Extensions journals. It collates all the information that can be known about the transaction from the available IMS Connect Extensions journal records. This information might include the client ID, IP address, overall performance, timings of significant events, IMS Connect exit specific performance, and dozens of additional pieces of information. You can use this index as both a reduced form of the IMS Connect Extensions journal and as input to IMS Problem Investigator and IMS Performance Analyzer.

Consider a scenario where all you know is the IP address, the approximate time of the problem, and the nature of the problem (say, “long delay”). Using only the IMS log, and with only limited details of the problem transaction, pinpointing the transaction can be difficult in itself, over and above examining the underlying cause of the problem itself.

Figure 3-13 shows an IMS Connect Transaction Index, which is merged with an IMS login IMS Problem Investigator.



Code	Description	Date	Time (LOCAL)
CA20	Connect Transaction TranCode=PRT3	2011-10-28	10.26.45.252647
01	Input Message TranCode=PART Source=Connect	2011-10-28	10.26.46.820236
35	Input Message Enqueue TranCode=PART	2011-10-28	10.26.46.820264
08	Application Start TranCode=PART Region=0001	2011-10-28	10.26.46.820754
5607	Start of UOR Program=DFSSAM02 Region=0001	2011-10-28	10.26.46.820755
31	DLI GU TranCode=PART Region=0001	2011-10-28	10.26.46.820791
5616	Start of protected UOW Region=0001	2011-10-28	10.26.46.821478
5E	SB Handler requests Image Capture Region=0001	2011-10-28	10.26.46.841470
5E	SB Handler requests Image Capture Region=0001	2011-10-28	10.26.46.841475
03	Output Message Response LTerm=3801 Source=Connect	2011-10-28	10.26.46.845009
31	Message GU for APPC LTerm=3801	2011-10-28	10.26.46.845032
33	Free Message	2011-10-28	10.26.46.845083
5610	Syncpoint Start of Phase 1 Region=0001	2011-10-28	10.26.46.847018
3730	Syncpoint End of Phase 1 Region=0001	2011-10-28	10.26.46.847074
33	Free Message	2011-10-28	10.26.46.847145
5612	Syncpoint End of Phase 2 Program=DFSSAM02 Region=0001	2011-10-28	10.26.46.847197
07	Application Terminate TranCode=PART Region=0001	2011-10-28	10.26.46.847830
5E	SB Handler requests Image Capture Region=0001	2011-10-28	10.26.46.847935
0901	SB Summary Statistics for PST	2011-10-28	10.26.46.847940
03	Output Message Response LTerm=3801 Source=Connect	2011-10-28	10.26.47.458046
31	Message GU for APPC LTerm=3801	2011-10-28	10.26.47.458072
33	Free Message	2011-10-28	10.26.47.460059
5610	Syncpoint Start of Phase 1 Region=0002	2011-10-28	10.26.47.460094
3730	Syncpoint End of Phase 1 Region=0002	2011-10-28	10.26.47.460146
33	Free Message	2011-10-28	10.26.47.460218
5612	Syncpoint End of Phase 2 Program=DFSSAM33 Region=0002	2011-10-28	10.26.47.460283
07	Application Terminate TranCode=PRT3 Region=0002	2011-10-28	10.26.47.461009
5E	SB Handler requests Image Capture Region=0002	2011-10-28	10.26.47.461127
0901	SB Summary Statistics for PST	2011-10-28	10.26.47.461132
CA20	Connect Transaction TranCode=PART	2011-10-28	10.26.47.856639
01	Input Message TranCode=PART Source=Connect	2011-10-28	10.26.47.857495
35	Input Message Enqueue TranCode=PART	2011-10-28	10.26.47.857522
08	Application Start TranCode=PART Region=0003	2011-10-28	10.26.47.857982
5607	Start of UOR Program=DFSSAM02 Region=0003	2011-10-28	10.26.47.857982
31	DLI GU TranCode=PART Region=0003	2011-10-28	10.26.47.858017
5616	Start of protected UOW Region=0003	2011-10-28	10.26.47.858634

Figure 3-13 IMS Connect transaction index that is merged in IMS Problem Investigator

Looking at the merged log, it is difficult to know exactly where the problem transaction occurred. However, you can filter the IMS Connect Transaction Index records to identify only those transactions matching the client's characteristics and then use tracking to pick up information from IMS about that specific transaction.

Figure 3-14 shows a filter for the IMS Connect Transaction Index.

```
File  Menu  Edit  Object Lists  Help
-----
Command ==> _____ Conditions _____ Row 1 to 2 of 2
Scroll ==> CSR

Code: CA20 Connect Transaction

/ Field Name +      Oper Value +
  RESPTIME          GT  2.0
  IPADDRESS         EQ  '172.17.69.25'
***** Bottom of data *****
```

Figure 3-14 Search for long transactions from an IP address

Figure 3-15 shows a possible result of this filtering. In this case, you can see a record of the problem transaction, its overall performance, and other key indicators.

```
File  Menu  Edit  Mode  Navigate  Filter  Time  Labels  Options  Help
-----
BROWSE  CEX000.QADATA.SLOWPRT3.ICON.D111028.IPIID  Record 00008119 More: < >
Command ==> _____ Scroll ==> CSR
Forwards / Backwards . . 00.00.00.000100 Time of Day . . 08.28.20.388036
Code Description          Date 2011-10-28 Friday Time (LOCAL)
/ -----
CA20 Connect Transaction 10.27.17.944277
TranCode=PRT3 Userid=CEX1 IMSID=ICDH ClientID=CLNAC13 Port=3801
LogToken=C8962E98210EAA00 SSN=0905 Response=2.163217 CM=1 SYNCLEVEL=1
TOV=38_MIN Socket=Pers
***** Bottom of Data *****
```

Figure 3-15 Initiate tracking from the transaction index that is identified by the filter

As shown in Figure 3-16 on page 101, tracking reveals all the IMS logs that are associated with the TCP/IP transaction. You can use the elapsed time view (on the right side) to identify significant timing delays and then drill down to the individual records for further investigation.

File Menu Edit Mode Navigate Filter Time Labels Options Help			
BROWSE CEX000.QADATA.SLOWPRT3.ICON.D111028.IPIID		Tracking active	
Command ==>		Scroll ==> CSR	
Forwards / Backwards . . 00.00.00.000100		Time of Day . . 08.28.20.388036	
Code Description		Date 2011-10-28 Friday	Time (Elapsed)
/			
CA20	Connect Transaction TranCode=PRT3		10.27.17.944277
01	Input Message TranCode=PRT3 Source=Connect		0.000713
35	Input Message Enqueue TranCode=PRT3		0.000032
08	Application Start TranCode=PRT3 Region=0001		0.000451
5607	Start of UOR Program=DFSSAM33 Region=0001		0.000000
31	DLI GU TranCode=PRT3 Region=0001		0.000036
5616	Start of protected UOW Region=0001		0.000650
5E	SB Handler requests Image Capture Region=0001		0.013011
5E	SB Handler requests Image Capture Region=0001		0.000005
03	Output Message Response LTerm=3801 Source=Connect		2.143720
31	Message GU for APPC LTerm=3801		0.000026
33	Free Message		0.002149
5610	Syncpoint Start of Phase 1 Region=0001		0.000068
3730	Syncpoint End of Phase 1 Region=0001		0.000053
33	Free Message		0.000071
5612	Syncpoint End of Phase 2 Program=DFSSAM33 Region=0001		0.000065
07	Application Terminate TranCode=PRT3 Region=0001		0.000787
***** Bottom of Data *****			

Figure 3-16 Both the IMS and IMS Connect log records for the transaction (tracked from index)

The result is that you have a clearer picture of exactly when the problem occurred, how it progressed, and where it was delayed in IMS.

3.2.3 Client fails to ACK

When a client fails to ACK/NAK a message, the symptoms can vary depending on the client's actions. IMS Connect Extensions events help you identify the problem by tracking the event flow.

In this example, you have a transaction with commit mode 1, sync level confirm, and transaction socket type. Instead of sending an acknowledgement of the message, the client closes the socket.

Figure 3-17 shows the sequence of events that are obtained through the IMS Problem Investigator ISPF interface. It shows the output message sending and the client response. You can also see a session error event.

BROWSE IMPOT00.WORKSHOP.CASE2.ICONLOG.TRACE		Record 00000005	More: < >
Command ==>		Scroll ==> CSR	
Forwards / Backwards . . 00.00.00.000100		Time of Day . . 08.28.20.388036	
Code	Description	Date 2007-10-12 Friday	Time (LOCAL)

0049	READ Socket		15.11.29.051635
00A4	Event Collection IRM Trace		15.11.29.051664
003D	Message Exit called for READ		15.11.29.051668
00A3	Event Collection OTMA Trace		15.11.29.051708
003E	Message Exit returned from READ TranCode=PART		15.11.29.051712
00A3	Event Collection OTMA Trace		15.11.29.051887
0041	Message sent to OTMA Type=Transaction		15.11.29.051894
00A3	Event Collection OTMA Trace		15.11.29.243377
0042	Message received from OTMA Type=Data		15.11.29.243395
00A3	Event Collection OTMA Trace		15.11.29.243591
00A3	Event Collection OTMA Trace		15.11.29.243595
003D	Message Exit called for XMIT		15.11.29.243598
00A6	Event Recording EXIT Output Message Trace		15.11.29.243633
003E	Message Exit returned from XMIT		15.11.29.243637
004A	WRITE Socket		15.11.29.243846
0049	READ Socket		15.11.29.244299
0049	READ Socket		15.11.29.244341
00A4	Event Collection IRM Trace		15.11.29.244362
003D	Message Exit called for READ		15.11.29.244381
00A3	Event Collection OTMA Trace		15.11.29.244412
003E	Message Exit returned from READ TranCode=DSPALLI		15.11.29.244416
003D	Message Exit called for EXER		15.11.29.245772
003E	Message Exit returned from EXER		15.11.29.245797
004A	WRITE Socket		15.11.29.245907
0047	Session Error		15.11.29.245915
000C	Begin CLOSE Socket		15.11.29.245957
00A3	Event Collection OTMA Trace		15.11.29.246018
0041	Message sent to OTMA Type=Response, Resp=NACK		15.11.29.246025
000D	End CLOSE Socket		15.11.29.246171
0048	Trigger Event for CLOSE		15.11.29.246191
***** Bottom of Data *****			

Figure 3-17 Session error in IMS Problem Investigator logs

The message exit receives the message from OTMA and constructs the output to send to the client. Therefore, in the event trace for the exit, you find relevant information. Figure 3-18 shows the formatted records for the session error.

```

File  Menu  Format  Help
BROWSE      IMPOT00.WORKSHOP.CASE2.ICONLOG.TRACE  Record 00000031 Line 00000000
Command ==>                                     Scroll ==> CSR
Form ==>      + _ Use Form in Filter              Format ==> FORM
***** Top of data *****
+0004 Code... 0047 Session Error
+00B0 STCK... C1556B63B6ADB441      LSN.... 000000000000012C
      Date... 2007-10-12 Friday      Time... 15.11.29.245915.265

+0000 CERE_47_LL..... 00C0
+0002 CERE_47_ZZ..... 0000
+0004 CERE_47_RECID..... A0 CERE_47_EVTID..... 47
+0006 CERE_47_PFXLL..... 0014
+0008 CERE_47_EFLAG..... 00 CERE_47_VER#..... 12
+000A CERE_47_TASKID..... ID of task recording event
+000A CERE_47_COL#..... 01 CERE_47_TKS#..... 04
+000C CERE_47_EVKEY..... C1556B63872BE542
+0014 CERE_47_VAR_LL..... 009C
+0016 CERE_47_VAR_APAR... 0001
+0018 CERE_47_VAR_FLAG3..... 80
+001A CERE_47_VAR_MSG.... 134 byte message area
+0000 004C0000 C8E6E2D7 F1F4F9F5 C540D6E3 *.<.HWSP1495E OT*
+0010 D4C140D7 D9D6E3D6 C3D6D340 E5C9D6D3 *MA PROTOCOL VIOL*
+0020 C1E3C9D6 D55E40D9 7EF2F06B 40C37EC3 *ATION; R=20, C=C*
+0030 D3D5C1C3 F1F0406B 40C4E27E C9D4C4F4 *LNAC10 , DS=IMD4*
+0040 40404040 6B40D47E E2C4D9C3 00000000 * , M=SDRC....*
+0050 00000000 00000000 00000000 00000000 *.....*
+0060 00000000 00000000 00000000 00000000 *.....*
+0070 00000000 00000000 00000000 00000000 *.....*
+0080 00000000 0000      *.....*
+00A0 CERE_47_VAR_SESRN..... 'WRITE '
+00A8 CERE_47_VAR_TOKEN..... 0000000000000000
***** End of data *****

```

Figure 3-18 Protocol violation error appearing in the log record

3.2.4 OTMA timeouts

Both the client and OTMA must set a timeout for requests, responses, and acknowledgements (ACK or NAK). Timeouts that are too long can lead to clients needlessly hanging or IMS processing requests for clients that never respond. Typically, a timeout needs to be set relatively aggressive so that in abnormal situations processing terminates rather than leaving the request hanging. However, if you set the timeout too aggressively, you create unnecessary network traffic as requests that would have otherwise completed must be rerun. Therefore, it is important to keep monitoring timeouts.

Figure 3-19 shows such a report in IMS Performance Analyzer.

IMS Performance Analyzer 4.3					
IMS Connect Exception List					
			Report from 240ct2008 17.14.02.86	Page 1	
Event Time	System	ID Description	Information	Start Time	
17.14.02.867096	IADH	11 Datastore Un-Available			
17.14.32.393973	IADH	2D Datastore status - Normal	Tmember=XCFMIADH DS=IMSC Status=0003		
17.14.32.394020	IADH	2D Datastore status - Normal	Tmember=XCFMIADH DS=IMS1 Status=0003		
17.26.02.620158	IADH	2D Datastore status - Warning	Tmember=XCFMIADH DS=IMSC Status=0002		
17.26.04.620242	IADH	45 OTMA time-out	Key=C33935502F10CA83 TOV=29	17.26.02.618972	
17.33.44.054865	IADH	2D Datastore status - Severe	Tmember=XCFMIADH DS=IMSC Status=0001		
17.33.46.054282	IADH	45 OTMA time-out	Key=C33937083DEB0943 TOV=29	17.33.44.051886	
17.33.47.058685	IADH	42 Msg from OTMA response is NAK	Key=C33937081B7D7401 DS=XCFMIADH Tpipe=7901 RSN=0030	17.33.47.057745	
17.33.50.076218	IADH	47 Session error	Key=C339370EF0DF1941 Type=READ	17.33.50.076206	
17.33.50.082014	IADH	47 Session error	Key=C339369868312A63 Type=READFAIL	17.33.50.080570	
17.40.26.595914	IADH	2D Datastore status - Normal	Tmember=XCFMIADH DS=IMSC Status=0003		
17.41.36.658482	IADH	47 Session error	Key=C33938CBDF9A5301 Type=READ	17.41.36.658469	
IMS Performance Analyzer 4.2					
IMS Connect Exception Summary - IADH					
			From 240ct2008 17.14.02.86 To 240ct2008 17.43.48.86	Page 16	
ID Description			Total		
11 Datastore Un-Available			1		
2D Datastore status - Severe			1		
2D Datastore status - Warning			1		
2D Datastore status - Normal			3		
42 Msg from OTMA response is NAK			1		
45 OTMA time-out			2		
47 Session error			3		

Figure 3-19 Reporting timeouts in IMS Performance Analyzer

You can also specifically look for timeouts in IMS Problem Investigator and identify what timeouts are used for requests that end up timing out, as shown in Figure 3-20.

004A	WRITE Socket	08.02.45.317115
0049	READ Socket	08.02.45.317491
0049	READ Socket	08.02.45.317542
00A4	Event Collection IRM Trace	08.02.45.317567
003D	Message Exit called for READ	08.02.45.317571
00A3	Event Collection OTMA Trace	08.02.45.317600
003E	Message Exit returned from READ TranCode=JLMTRAN1	08.02.45.317604
00A3	Event Collection OTMA Trace	08.02.45.317662
0041	Message sent to OTMA Type=Response, Resp=ACK	08.02.45.317672
S 0045	OTMA Time-out	08.02.47.319014
00A3	Event Collection OTMA Trace	08.02.47.319109
003D		9115
00A6	+0004 Code... 0045 OTMA Time-out	9156
003E	+001C STCK... C15A1307375E6760 LSN.... 0000000000000036	9159
004A	Date... 2007-10-16 Tuesday Time... 08.02.47.319014.460	9365
000C		9404
000D	+0000 CERE_45_LL..... 002C	9616
0048	+0002 CERE_45_ZZ..... 0000	9644
	+0004 CERE_45_RECID..... A0 CERE_45_EVTID..... 45	
	+0006 CERE_45_PFXLL..... 0014	
	+0008 CERE_45_EFLAG..... 00 CERE_45_VER#..... 12	
	+000A CERE_45_TASKID..... ID of task recording event	
	+000A CERE_45_COL#..... 01 CERE_45_TKS#..... 04	
	+000C CERE_45_EVKEY..... C15A130525EAF701	
	+0014 CERE_45_VAR_LL..... 0008	
	+0016 CERE_45_VAR_APAR..... 0001	
	+001B CERE_45_VAR_TOV.... 29	
	+001B CERE_45_VAR_TOV.... 29	
	Timeout value: 2 seconds	

Figure 3-20 Identifying the timeout value from the IMS Connect Extensions journal

As shown in Figure 3-21, if the timeout must be adjusted, you can change this timeout dynamically for all clients that use a specific transaction code by overriding the transaction timer in IMS Connect Extensions.

```

Name . . . . : JLMTRAN1
Description . . Transaction with slow delay

Application . . . DEFAPP +

/ Override Transaction Timer      Message timeout . . 00 (default)
                                ACK/NAK timeout . . 63 (1 minute)
_ Override Transaction Expiration Set F1_TRNEXP . . . 1 1. On
                                2. Off
/ Override Client ID Cancellation Set F3_CANCID . . . 1 1. On
                                2. Off
_ Activate Transaction Routing
  _ Override Application options
    Route transactions to:
    - 1. All Datastores
      2. Datastore . . . . . _____ +
      3. Datastore Group . . . _____ +
      4. Affinity List . . . . _____ +

    Routing Error processing:
    - 1. Use the original datastore in the message request
      2. Reject the transaction
  
```

Figure 3-21 Overriding the ACK/NAK timeout for a transaction code

3.2.5 Duplicate clients

Another common error condition is the duplicate client. This condition can happen when a client attempts to connect using a client ID of an active session, when Client ID cancellation is not set.

Figure 3-22 shows this scenario in IMS Problem Investigator.

```

+0004 Code... 0047 Session Error
+00B0 STCK... C15564202486BD20      LSN.... 00000000000000AB
      Date... 2007-10-12 Friday      Time... 14.38.59.344491.820

+0000 CERE_47_LL..... 00C0
+0002 CERE_47_ZZ..... 0000
+0004 CERE_47_RECID..... A0 CERE_47_EVTID..... 47
+0006 CERE_47_PFXLL..... 0014
+0008 CERE_47_EFLAG..... 00 CERE_47_VER#..... 12
+000A CERE_47_TASKID..... ID of task recording event
+000A CERE_47_COL#..... 01 CERE_47_TKS#..... 04
+000C CERE_47_EVKEY..... C155642024335862
+0014 CERE_47_VAR_LL..... 009C
+0016 CERE_47_VAR_APAR... 0001
+0018 CERE_47_VAR_FLAG3..... 80
+001A CERE_47_VAR_MSG.... 134 byte message area
      +0000 E6E3D6F6 40404040 00000000 00000000 *WTO6 .....*
      +0010 C8E6E2E2 F0F7F4F2 F3F8F0F1 40404040 *HWSS07423801 *
      +0020 C1E3D4F0 F0F14040 C9D4C4F4 40404040 *ATM001 IMD4 *
      +0030 00000008 00000000 C4E4D7C5 C3D3D5E3 *.....DUPECLNT*
      +0040 E2D9C5F4 40404040 00000000 00000000 *SRE4 .....*
      +0050 00000000 00000000 00000000 00000000 *.....*
      +0060 00000000 00000000 00000000 00000000 *.....*
      +0070 00000000 00000000 00000000 00000000 *.....*
      +0080 00000000 0000 *.....*
+00A0 CERE_47_VAR_SESRN..... C4E4D7C516489000
+00A8 CERE_47_VAR_TOKEN..... 0000000000000000

```

Figure 3-22 Duplicate client error in IMS Connect Extensions journal

Just as you did with Transaction time override, you can also set Client ID cancellation in IMS Connect Extensions to mitigate this issue, as shown in Figure 3-23.

```

Name . . . . : JLMTRAN1
Description . . Transaction with slow delay

Application . . . DEFAPP +

/ Override Transaction Timer      Message timeout . . 00 (default)
                                ACK/NAK timeout . . 63 (1 minute)
_ Override Transaction Expiration Set F1_TRNEXP . . . 1 1. On
                                2. Off
/ Override Client ID Cancellation Set F3 CANCEID . . . 1 1. On
                                2. Off
_ Activate Transaction Routing
  _ Override Application options
    Route transactions to:
    - 1. All Datastores
    - 2. Datastore . . . . . _____ +
    - 3. Datastore Group . . . _____ +
    - 4. Affinity List . . . . _____ +

    Routing Error processing:
    - 1. Use the original datastore in the message request
    - 2. Reject the transaction

```

Figure 3-23 Overriding the Client ID cancellation option for a transaction code

3.2.6 Open database issues

This section shows how you can use the IMS Tools to identify and resolve common problems with the sample applications. You can classify Open Database issues into the following categories:

- ▶ Session errors: Conditions that generate distinct errors, for example, specifying the wrong alias name or trying to access a stopped PSB.
- ▶ Performance problems: IMS provides an output, but the processing time is slow.
- ▶ Unexpected responses: The client receives information from IMS, but it is not the feedback that the client was expecting.

This chapter has shown that IMS Problem Investigator can be used to examine the requests that the client sends, how they are processed by IMS, and the replies that are returned, allowing you to understand the context of unexpected responses. The next two sections describe session errors and performance problems.

Session errors

When a session error occurs, IMS Connect Extensions writes an A047 record to the journal. You can use IMS Problem Investigator filtering to identify any session errors in the journal. Figure 3-24 shows a simple filter that shows only A047 records in the journal.

File	Menu	Edit	Mode	Navigate	Filter	Time	Labels	Options	Help

BROWSE	CEX000.QADATA.REDBOOK.ERR01.ICON.D100331							Tracking inactive	
Command ==>								Scroll ==> CSR	
Forwards / Backwards . .				00.00.00.000100		Time of Day . .		16.46.22.845746	
Code Description				Date 2010-03-31		Wednesday		Time (LOCAL)	
/	-----								
A047 Session Error								11.54.55.710442	
A047 Session Error								12.03.02.490039	
A047 Session Error								12.32.09.670281	
***** Bottom of Data *****									

Figure 3-24 Filtering by A047 records

The log record itself often contains enough information to understand the cause of the problem. For example, as shown in Figure 3-25, if you select the last session error, you can see the message area and discover that the client requested RRS when RRS was not available.

```

File  Menu  Format  Help
-----
BROWSE      CEX000.QADATA.REDBOOK.ERR01.ICON.D100 Record 00000344 Line 00000010
Command ==>                                     Scroll ==> CSR
Form      ==>          +      Use Form in Filter          Format ==> FORM
+000A CERE_47_TASKID..... ID of task recording event
+000A CERE_47_COL#..... 01 CERE_47_TKS#..... 04
+000C CERE_47_EVKEY..... C5C21834F1A8FC61
+0014 CERE_47_VAR_LL..... 009C
+0016 CERE_47_VAR_APAR... 0001
+0018 CERE_47_VAR_FLAG3..... 80
+001A CERE_47_VAR_MSG.... 134 byte message area
      +0000 00640000 C8E6E2D2 F2F8F8F0 C540D9D9      *...HWSK2880E RR*
      +0010 E240C3D6 D4D4C1D5 C440C6C1 C9D3C5C4      *S COMMAND FAILED*
      +0020 5E40C37E D6C4C2F2 C5F4F2F3 6B40C3D7      *; C=0DB2E423, CP*
      +0030 7EE2E8D5 C3C3E3D3 4040406B 40D77EF4      *=SYNCCTL   , P=4*
      +0040 F8F8F5F5 4040406B 40D97EF0 F0F0F46B      *8855   , R=0004,*
      +0050 40D9E27E D9D9E2D5 C1E5C9D3 6B40D47E      * RS=RRSNAVIL, M=*
      +0060 D4D9C3E5 00000000 00000000 00000000      *MRCV.....*
      +0070 00000000 00000000 00000000 00000000      *.....*
      +0080 00000000 0000                                *.....*
+00A0 CERE_47_VAR_SESRN..... 'WRITE   '
+00A8 CERE_47_VAR_TOKEN..... 0000000000000000
***** End of data *****

```

Figure 3-25 Displaying the message area

In some cases, you might be interested in examining the complete flow of a request that generated a session error. In such cases, you can use tracking (TX line action) to reveal all of the event records that are associated with a particular session error, as shown in Figure 3-26 on page 109.

In this case, you can see the progression of the request leading up to the session error. The client completes security authentication, but when the Access RDB request is made, triggering the allocation of the PSB, the request then receives an RDB Not Found DRDA object, which triggers the session error.

```

File  Menu  Edit  Mode  Navigate  Filter  Time  Labels  Options  Help
-----
BROWSE      CEX000.QADATA.REDBOOK.ERR01.ICON.D100331      Record 00000013 More: < >
Command ==>                                         Scroll ==> CSR
Forwards / Backwards . . 00.00.00.000100      Time of Day . . 16.46.22.845746
Code Description                                Date 2010-03-31 Wednesday Time (Relative)
/  ----  -----
TX A049 READ Socket                                -0.256794
A05B DRDA 106E SECCHK-Security Check              -0.256786
A063 ODBM Security Exit called                     -0.256755
A064 ODBM Security Exit returned                   -0.256668
A05C DRDA 1219 SECCHKRM-Security Check Reply Message -0.256594
A04A WRITE Socket                                -0.256516
A049 READ Socket                                -0.000293
A049 READ Socket                                -0.000223
A05B DRDA 2001 ACCRDB-Access RDB                   -0.000216
A05D ODBM begin Allocate PSB (APSB) Program=AUTPSB11 -0.000194
A061 ODBM Routing Exit called                     -0.000185
A062 ODBM Routing Exit returned                   -0.000033
A05C DRDA 2211 RDBNFNRM-RDB Not Found              11.54.55.710366
A04A WRITE Socket                                +0.000064
A047 Session Error                                +0.000075
A00C Begin CLOSE Socket                            +0.000108
A00D End CLOSE Socket                              +0.000325

```

Figure 3-26 Tracking the complete flow

Performance problems

Using IMS Problem Investigator, you can map the relative timing of an Open Database request and broadly discover how much time the requests spend in various stages of processing:

- Client
- IMS Connect
- ODBM
- IMS

The log records that are available for IMS (often the area where critical insight is required) are similar to those that are available for CICS DBCTL transactions. As such, they provide only limited information about the timing of the DL/I calls themselves. If the requests are predominantly queries and not updates, little information is available in the IMS logs.

To mitigate this situation, use database trace, such as the IMS Monitor. However, the most detailed trace is available from the OMEGAMON for IMS on z/OS application trace facility (ATF). Using ATF, you can record DL/I call characteristics. IMS Tools allow you to connect between the DL/I call activity and the DRDA requests that are being passed by the client, as shown in Figure 3-27.

Figure 3-27 shows the Open Database requests and the DL/I calls themselves, including the duration of each call and the processor usage for each call.

File Menu Edit Mode Navigate Filter Time Labels Options Help			

BROWSE	CEX000.QADATA.REDBOOK.DRDAT111.ICON.D1003	Record 00000308	More: < >
Command ==>			Scroll ==> CSR
Forwards / Backwards	. . 00.00.00.000100	Time of Day . .	16.46.22.845746
Code Description		Date 2010-03-31 Wednesday	Time (LOCAL)

/			
A049	READ Socket		13.46.47.095038
A05B	DRDA CC06 SSALIST-List of segment search argument		13.46.47.095045
A0AA	ODBM Trace: Message sent to ODBM		13.46.47.095985
A069	Message sent to ODBM		13.46.47.096016
06	OSAM IWAIT start TranCode=ODBA02CD Region=0003		13.46.47.142891
20	Database Open Database=EMPDB2 Region=0003		13.46.47.143647
06	OSAM IWAIT start TranCode=ODBA02CD Region=0003		13.46.47.181506
20	Database Open Database=AUTODB Region=0003		13.46.47.182252
06	OSAM IWAIT start TranCode=ODBA02CD Region=0003		13.46.47.191442
01	DLI GHU Database=EMPLDB2 SC=' ' Elapse=0.095875		13.46.47.096570
B021	DLI Database Trace Database=EMPLDB2 Func=GHU		13.46.47.192378
A0AA	ODBM Trace: Message received from ODBM		13.46.47.192881
A06A	Message received from ODBM		13.46.47.192909
A05C	DRDA 2205 OPNQRYSRM-Open Query Complete		13.46.47.193186
A04A	WRITE Socket		13.46.47.193515
A048	Trigger Event for ODBMMSG		13.46.47.193554
A03C	Prepare READ Socket		13.46.48.120636

Figure 3-27 DRDA and DL/I flow

You can control the amount of information IMS Problem Investigator shows (to view extended details, scroll right by pressing F11). As shown in Figure 3-28, the ATF records provide information about the processor usage and elapsed time for each DLI request.

```

File  Menu  Edit  Mode  Navigate  Filter  Time  Labels  Options  Help
-----
BROWSE      CEX000.QADATA.REDBOOK.DRDAT111.ICON.D1003  Record 00000316 More: < >
Command ==>                                         Scroll ==> CSR
Forwards / Backwards . . 00.00.00.000100      Time of Day . . 16.46.22.845746
Code Description                                Date 2010-03-31 Wednesday  LSN
/
-----
06  OSAM IWAIT start                                2-000000000000004
    TranCode=ODBA02CD Program=AUTPSB11 Userid=AUTPSB11 Region=0003
    IMSID=ODBA02CD RecToken=ODBA02CD/0000000100000000 Elapse=0.000725
-----
01  DLI GHU                                2-000000000000005
    TranCode=ODBA02CD Program=AUTPSB11 Userid=AUTPSB11 Database=EMPLDB2
    Region=0003 IMSID=ODBA02CD RecToken=ODBA02CD/0000000100000000 SC=' '
    Elapse=0.095875 CPU=0.002190
-----
B021 DLI Database Trace                                3-000000000000105
     LTerm=EMPLDB2 Database=EMPLDB2 Region=0003
     RecToken=ODBA02CD/0000000100000000 Func=GHU Elapsed=00.095631
-----
A0AA ODBM Trace: Message received from ODBM          1-000000000000359
     IMSID=IBDE0D0D LogToken=C5C228E17B387162
-----
A06A Message received from ODBM                      1-00000000000035A

```

Figure 3-28 Displaying detailed information by pressing F11

IMS Problem Investigator correlates the ATF data with the Open Database request itself so that you can see exactly which request led to which DLI calls and what their timings were.

As shown in Figure 3-29, if you select the ATF generated 01 record, you can see the IO area that is used for the request itself.

```
File  Menu  Format  Help
-----
BROWSE      CEX000.QADATA.REDBOOK.DRDAT111.ICON.D Record 00000317 Line 00000042
Command ==>                               Scroll ==> CSR
Form       ==>           +      Use Form in Filter           Format ==> FORM
+00DC  ATRDXEL.... DL/I Trace Element
+00DC  ATRDX@E.... 14ABD400   ATRDXTY.... 02           ATRDXF..... 00
+00E2  ATRDX#..... 0008
+00E4  ATRDXV..... Element Data - Key Feedback Area
      +0000  F2F2F2F2 F2F2C3C1           *222222CA           *

+00EC  ATRDXEL.... DL/I Trace Element
+00EC  ATRDX@E.... 14AAD2E8   ATRDXTY.... 01           ATRDXF..... 00
+00F2  ATRDX#..... 004C
+00F4  ATRDXV..... Element Data - I/O Area
      +0000  F2F2F2F2 F2F2C299 96A69540 40404040   *222222Brown   *
      +0010  40404040 40404040 40404040 404040D9   *              R*
      +0020  96954040 40404040 40404040 40404040   *on            *
      +0030  40404040 40404040 E2D6D4C5 40E2E3D9   *              SOME STR*
      +0040  C5C5E340 40404040 40404040           *EET            *

+0140  ATRDXEL.... DL/I Trace Element
+0140  ATRDX@E.... 14AAD540   ATRDXTY.... 04           ATRDXF..... 00
+0146  ATRDX#..... 0036
```

Figure 3-29 Displaying the I/O area

3.2.7 Synchronous callout issues

Synchronous callout offers a powerful feature that allows IMS applications to call web services and receive a response. Because these requests are synchronous, monitoring and reporting response time characteristics is critical for the overall health of your IMS environment.

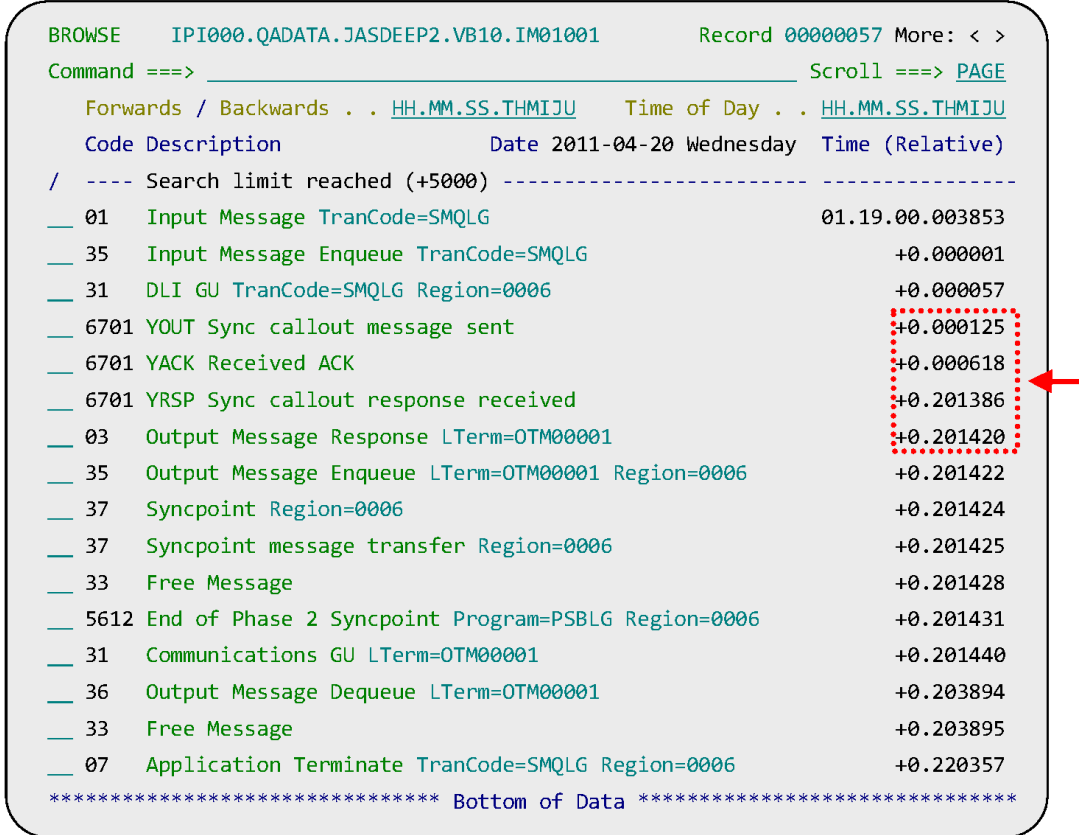
IMS Connect Extensions journals collect synchronous callout events and this information can also be used for reporting in IMS Problem Investigator and IMS Performance Analyzer.

In Figure 3-30, you can see that the callout response time is a large percentage of the overall processing time.

Transactions w/ synch-callout							
Data from 10.18.37 19Apr2011 to 10.21.06 19Apr2011							
Tran	Avg CPU	Avg InputQ	Avg Process	Avg SyncCout	Avg Total	Avg SyncCout	Avg SyncCout
Count	Time	Time	Time	RespTime	IMS Time	ACK Cnt	NAK Cnt
57760	0.0023	0.0052	0.5673	0.4723	0.5725	1	0

Figure 3-30 Synchronous callout reporting in IMS Performance Analyzer

As shown in Figure 3-31, you can then use IMS Problem Investigator to identify the request characteristics that caused the delay.



BROWSE IPI000.QADATA.JASDEEP2.VB10.IM01001 Record 00000057 More: < >
 Command ==> Scroll ==> PAGE
 Forwards / Backwards . . HH.MM.SS.THMIJU Time of Day . . HH.MM.SS.THMIJU
 Code Description Date 2011-04-20 Wednesday Time (Relative)

Code	Description	Time (Relative)
/	---- Search limit reached (+5000) -----	
01	Input Message TranCode=SMQLG	01.19.00.003853
35	Input Message Enqueue TranCode=SMQLG	+0.000001
31	DLI GU TranCode=SMQLG Region=0006	+0.000057
6701	YOUT Sync callout message sent	+0.000125
6701	YACK Received ACK	+0.000618
6701	YRSP Sync callout response received	+0.201386
03	Output Message Response LTerm=OTM00001	+0.201420
35	Output Message Enqueue LTerm=OTM00001 Region=0006	+0.201422
37	Syncpoint Region=0006	+0.201424
37	Syncpoint message transfer Region=0006	+0.201425
33	Free Message	+0.201428
5612	End of Phase 2 Syncpoint Program=PSBLG Region=0006	+0.201431
31	Communications GU LTerm=OTM00001	+0.201440
36	Output Message Dequeue LTerm=OTM00001	+0.203894
33	Free Message	+0.203895
07	Application Terminate TranCode=SMQLG Region=0006	+0.220357

***** Bottom of Data *****

Figure 3-31 Synchronous callout event flow in IMS Problem Investigator

3.2.8 OTMA flood condition

OTMA has a flood notification system that outputs a warning when data stores are in-flight. You can use the IMS Connect Extensions GUI to identify an OTMA flood condition and investigate its causes, as shown in Figure 3-32.

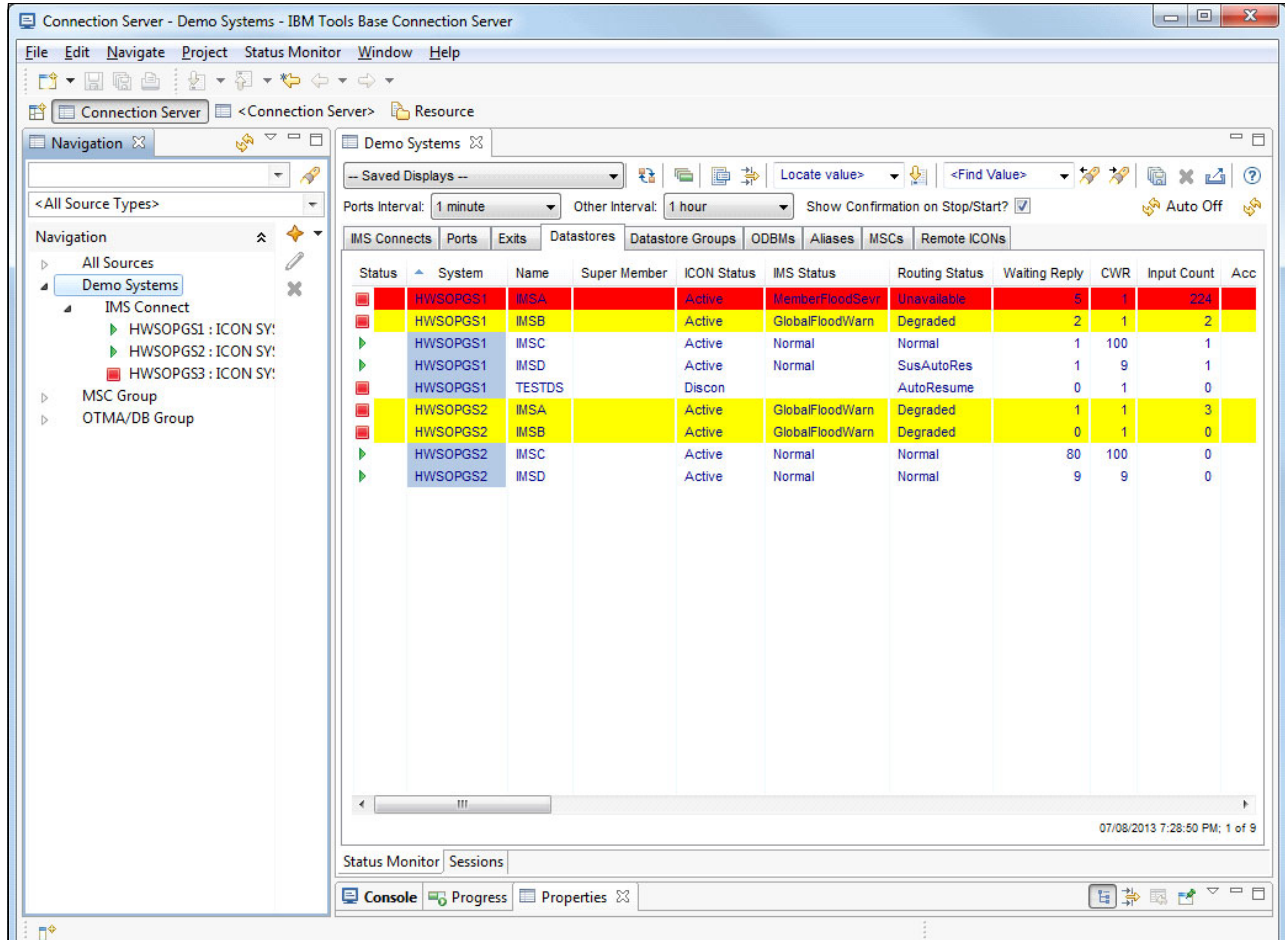


Figure 3-32 OTMA flood condition in the IMS Connect Extensions GUI

In Figure 3-32, the GUI highlights the data stores in various flood states.

Here are the corrective actions:

- ▶ Restart the PSB.
- ▶ Increase the flood threshold.
- ▶ Change capacity weights to help reduce the overall load.

In the example that is shown in Figure 3-33, we use IMS Connect Extensions' command facility to alter the flood thresholds.

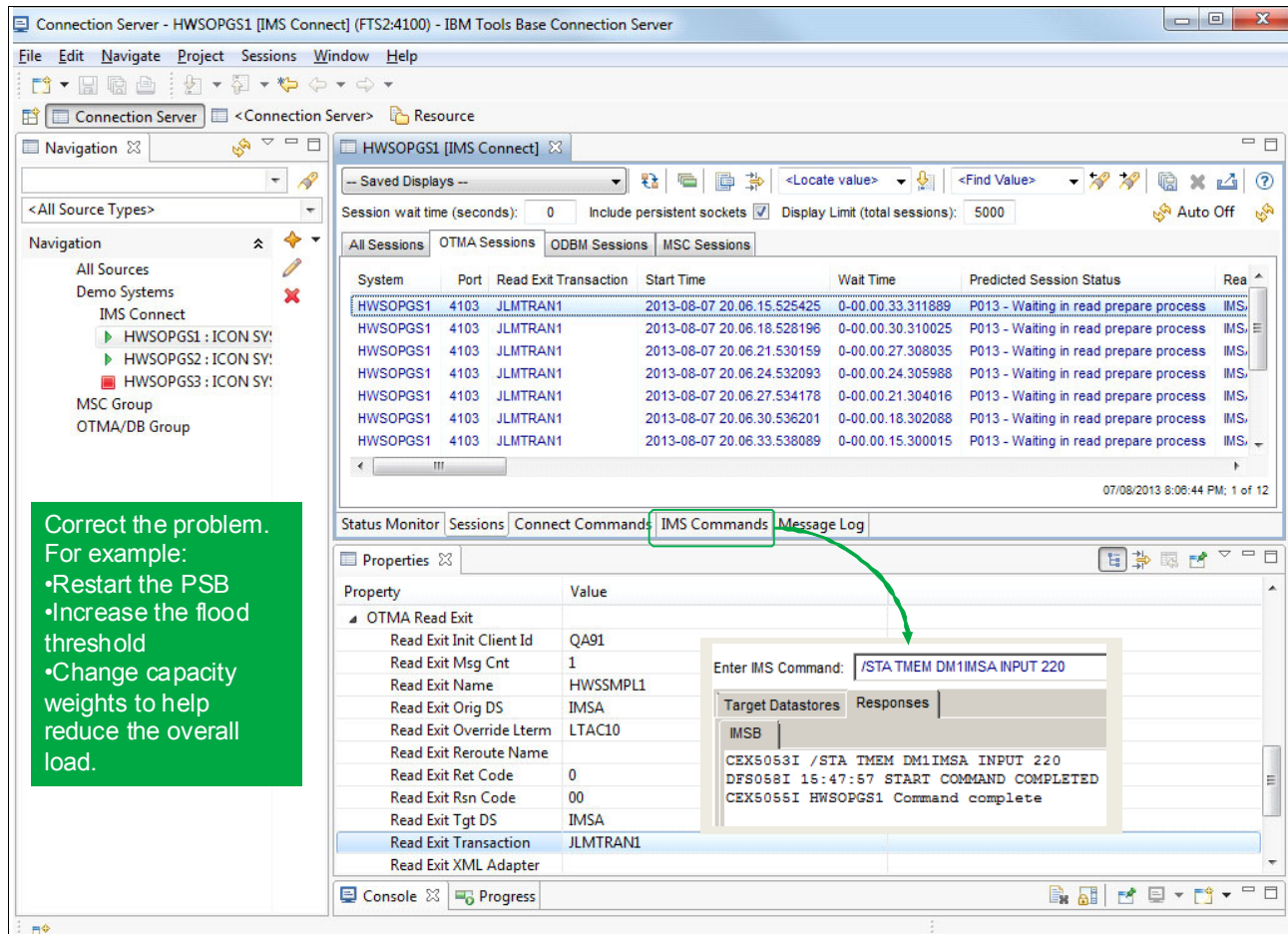


Figure 3-33 Issuing a command to increase the flood threshold from the IMS Connect Extensions GUI

3.3 Scheduled and unscheduled outages

This section describes some of the implications and considerations for managing IMS Connect to minimize the impact on clients from scheduled and unscheduled outages and to increase the resilience and efficiency of the IMS topology in response to outages. In this context, we look at the effect of IMS outages on existing TCP/IP sessions and the effect of IMS Connect outages have on persistent socket sessions, which are then not balanced by the SYSPLEX distributor. As shown in Figure 3-34, outages in different subsystems have a different impact on the wider topology and need different approaches in order to be dealt with.

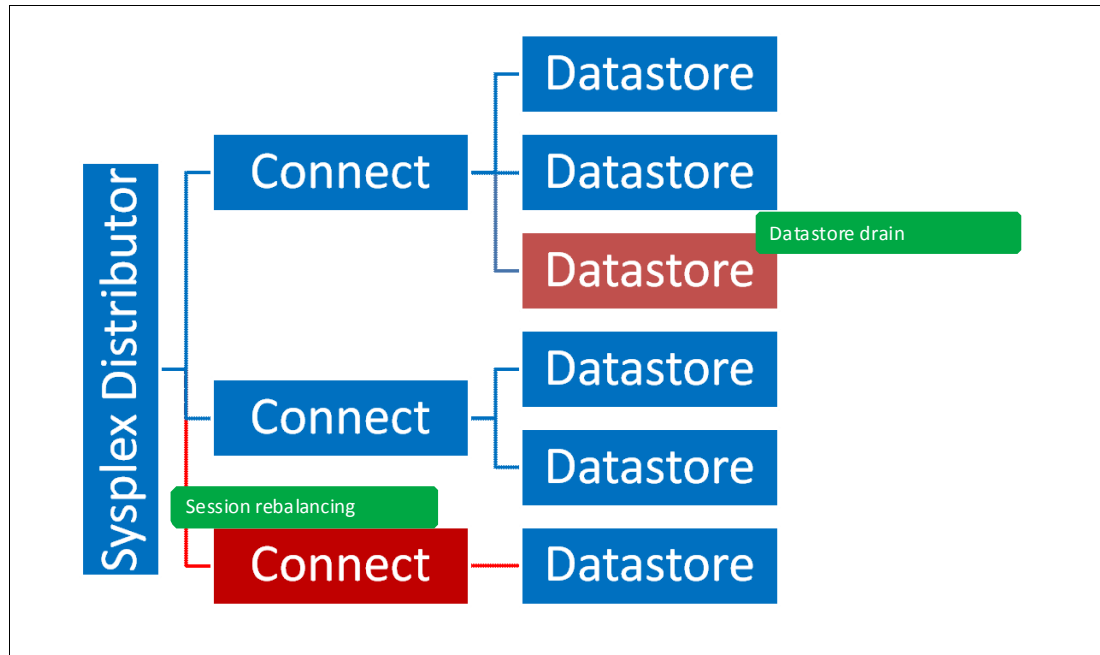


Figure 3-34 Outages at different subsystems require different responses and coordination

This section describes the following topics:

- ▶ Minimizing disruptions from outages
- ▶ Session rebalancing
- ▶ Centrally managing client options

3.3.1 Minimizing disruptions from outages

Consider the following problem: You must take IMS systems offline for maintenance, and routing helps you ensure that clients have alternative targets, but you cannot take the existing data store offline without potentially disrupting existing active sessions.

IMS Connect Extensions includes a data store drain command, which changes the status of the data store to Suspended: no new requests are routed to this system, but existing sessions are allowed to complete their processing.

Here are the steps in this process:

1. Issue the drain command against all data stores that associated with the IMS system. You can do this in ISPF, batch, or the GUI. If multiple data stores are associated with a single IMS, you can maintain lists of data stores and issue commands against the list.

The existing workload is allowed to complete.

2. Use ISPF, the GUI, or batch to monitor the number of outstanding requests on the data store until they drop to 0. Take IMS offline, perform the maintenance, and restore IMS.
3. Restore the data store.

Draining data stores through the GUI

In the following example, we perform this process through the GUI:

1. Click **Route Drain**. The status of the data store changes to Suspended. There are still two transactions waiting for a reply, as shown in Figure 3-35.

Status	System	Name	ICON Status	IMS Status	Routing Status	Waiting Reply	CWR	Accept
▶	HWSOPGS1	IMSA	Active	Normal	Normal	5	1	
▶	HWSOPGS1	IMSB	Active	Normal	Suspended	2	1	
▶	HWSOPGS1	IMSC	Active	Normal	Normal	1	100	
▶	HWSOPGS1	IMSD	Active	Normal	Normal	1	9	

Figure 3-35 Route Drain suspends the data store and prevent new activity but allows existing sessions to complete

2. When no sessions are waiting, right-click to stop the data store.
3. Perform maintenance.
4. Click to start the data store, as shown in Figure 3-36.

Status	System	Name	ICON Status	IMS Status	Routing Status	Waiting Reply	CWR	Accept
▶	HWSOPGS1	IMSA	Active	Normal	Normal	5	1	
▶	HWSOPGS1	IMSB	Active	Normal	Suspended	0	1	
▶	HWSOPGS1	IMSC	Active	Normal	Normal	1	100	
▶	HWSOPGS1	IMSD	Active	Normal	Normal	1	9	

Figure 3-36 Stop the data store when there are no sessions and start the data store when maintenance completes

Initiating a drain in batch

Batch controls allow you to stop and start data stores in batch, which allows you to incorporate data store drains with your existing automation processing, as shown in Figure 3-37.

```
SDSF EDIT      HWSREA      (JOB93678) JCLEEDIT      Columns 00001 00072
Command ==>      Scroll ==> CSR
***** Top of Data *****
...
000021  ROUTE ACTION=DRAIN,DATASTORE=datastore[, AUTORESUME]
...
000045  ROUTE ACTION=RESUME,DATASTORE=datastore
```

Figure 3-37 Initiate a data store drain in batch

Batch QUERY commands allow you to automate responses to changing data store conditions: processing status and number of active connections. You can query a specific data store or a collection (list). Lists can represent all the data stores that are associated with an IMS, as shown in Figure 3-38.

```
SDSF EDIT      HWSREA      (JOB93678) JCLEEDIT      Columns 00001 00072
Command ==>      Scroll ==> CSR
***** Top of Data *****
...
000021  QUERY TYPE=PENDING_RESPONSES DATASTORE=datastore|DSLIS=routinglist...
```

Figure 3-38 Query the status of a data store in batch

The response provides status and number of pending responses, as shown in Example 3-5.

Example 3-5 Output of data store batch query

```
CEX5133I QUERY DS(DS01): Routing Status is NORMAL with 8 responses pending.
CEX5133I QUERY DS(DS01): Routing Status is SUSPENDED with 0 responses pending.
CEX5133I QUERY DS(DS01): Routing Status is SUSPENDED with 2 responses pending.
CEX5133I QUERY DS(DS01): Routing Status is DEGRADED with 5 responses pending.
CEX5133I QUERY DS(DS01): Routing Status is UNAVAILABLE with 14 responses pending.
```

You can set custom thresholds for message rates and output messages when thresholds are reached for the data store, group, or IMS Connect.

3.3.2 Session rebalancing

Session rebalancing helps ensure that persistent socket sessions are balanced between IMS Connect systems in a Sysplex distributor environment. When the number of sessions on a persistent socket reaches this threshold, no further connections are allowed, and the persistent socket is closed when the last session ends.

You can set this option as part of the system definition process in IMS Connect Extensions by using the Session Message Limit, as shown in Figure 3-39.

```
File  Menu  Settings  Help

EDIT                                     System Definition
Command ===> _____

Name . . . . : ICOND00
Description . . Workshop demo system1

- Activate Access Control                Security app
- Activate PassTicket Generation

/ Activate Advanced Features
- Activate Pacing
  Interval count . . . . 3
  Warning threshold . . 0                Reject thres
- Activate Session Message Limit         Limit thresh
- Activate Security
```

Figure 3-39 Configuring session rebalancing

3.3.3 Centrally managing client options

The scenarios in this book have shown cases where you might want to centrally and dynamically alter client-set options, such as transaction expiration and client ID cancellation.

Here are more options that can be overridden:

- ▶ Message translation: Uses different code page translation than the defaults that are offered by the exits.
- ▶ Extended RSM: Provides more feedback to clients when an RSM appears, as shown in Figure 3-40.

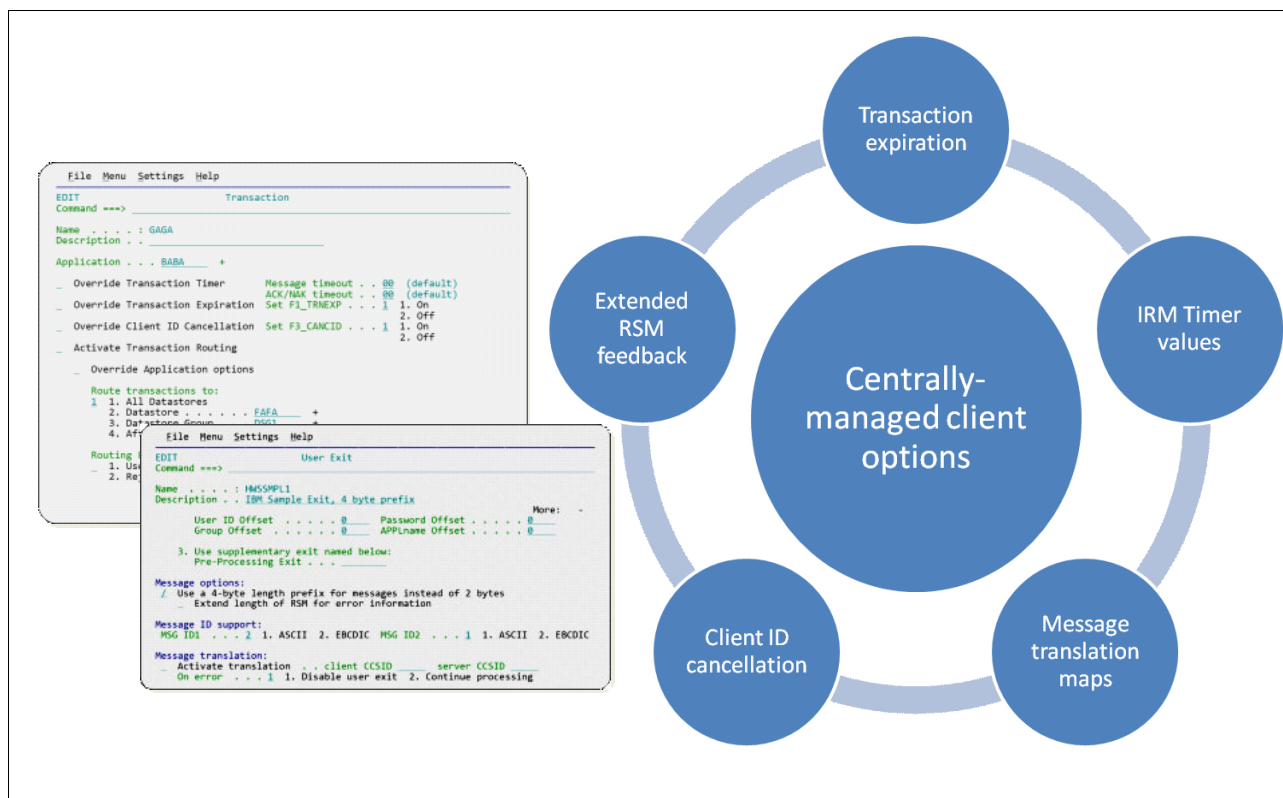


Figure 3-40 Centrally-configurable client options

3.4 Application development considerations

Consider a modern integrated developer environment (IDE) in the context of client/server development. From the perspective of the development environment, client/server development in modern IDEs, such as Rational Developer for System z, allows developers to control their own *sandbox* instances of application servers.

In Figure 3-41 on page 121, a developer has control over a local IBM WebSphere Liberty server.

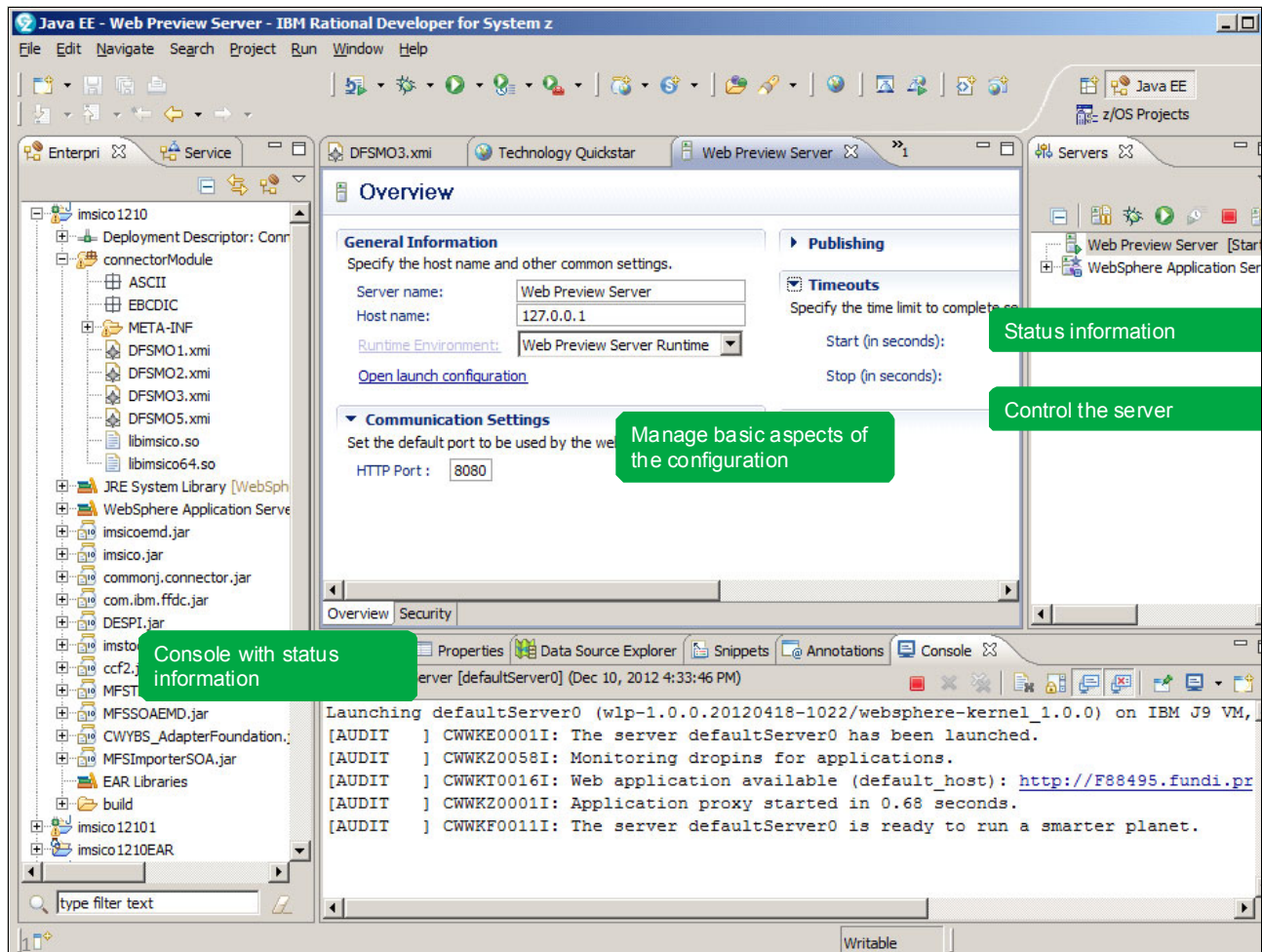


Figure 3-41 Common features for client/server development

From the IDE, the developer can perform basic server configuration (even without deep server administration knowledge), and they can view status information and control the server functions. If errors occur, they can see information about these errors directly in the console.

The developers can stop and start the Apache Tomcat instance, deploy their application to the server, profile their application on the server, and view basic debugging and logging from the server.

Consider a developer that is working on IMS applications. He needs a similar level of visibility and insight into both IMS and IMS Connect. Without some control and instrumentation, application development time is slowed and there is a greater reliance on IMS system programmers.

In Figure 3-42, you can see the same IDE with the IMS Connect Extensions plug-in. Using the plug-in, the developer can get basic status information for IMS Connect, can stop and start data stores and exit, and can run commands to perform additional debugging or make the required changes.

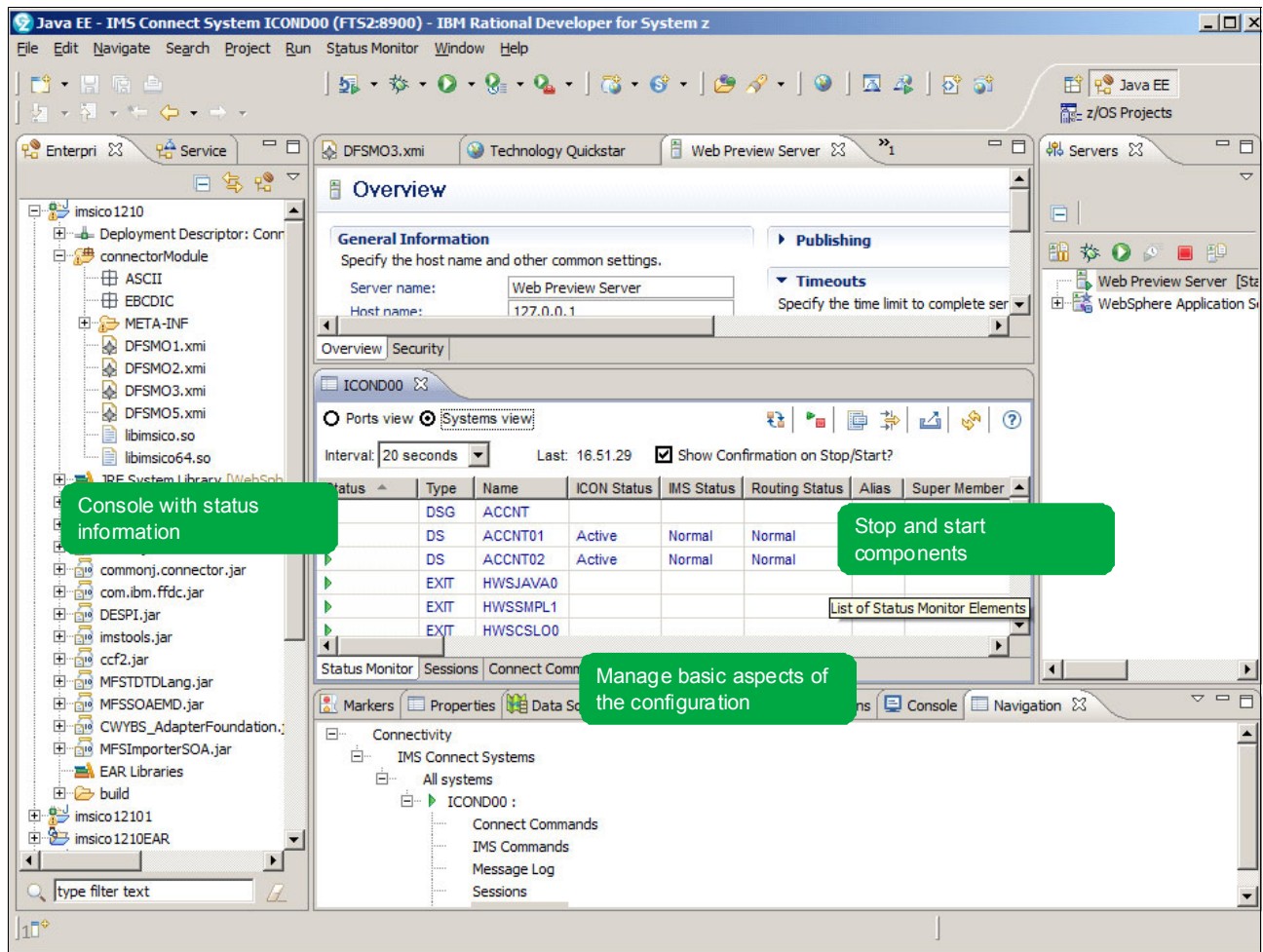


Figure 3-42 An IDE extended with IMS Connect Extensions

The IMS Connect Extensions GUI plug-in integrates with tools such as the IMS Explorer for Developers, providing developers a greater level of autonomy and control for application development.

In Figure 3-43, a developer is writing an SQL query that uses the IMS Open Database to retrieve an output. In many ways, this environment is similar to the one that they use to develop applications that get data from, for example, DB2.

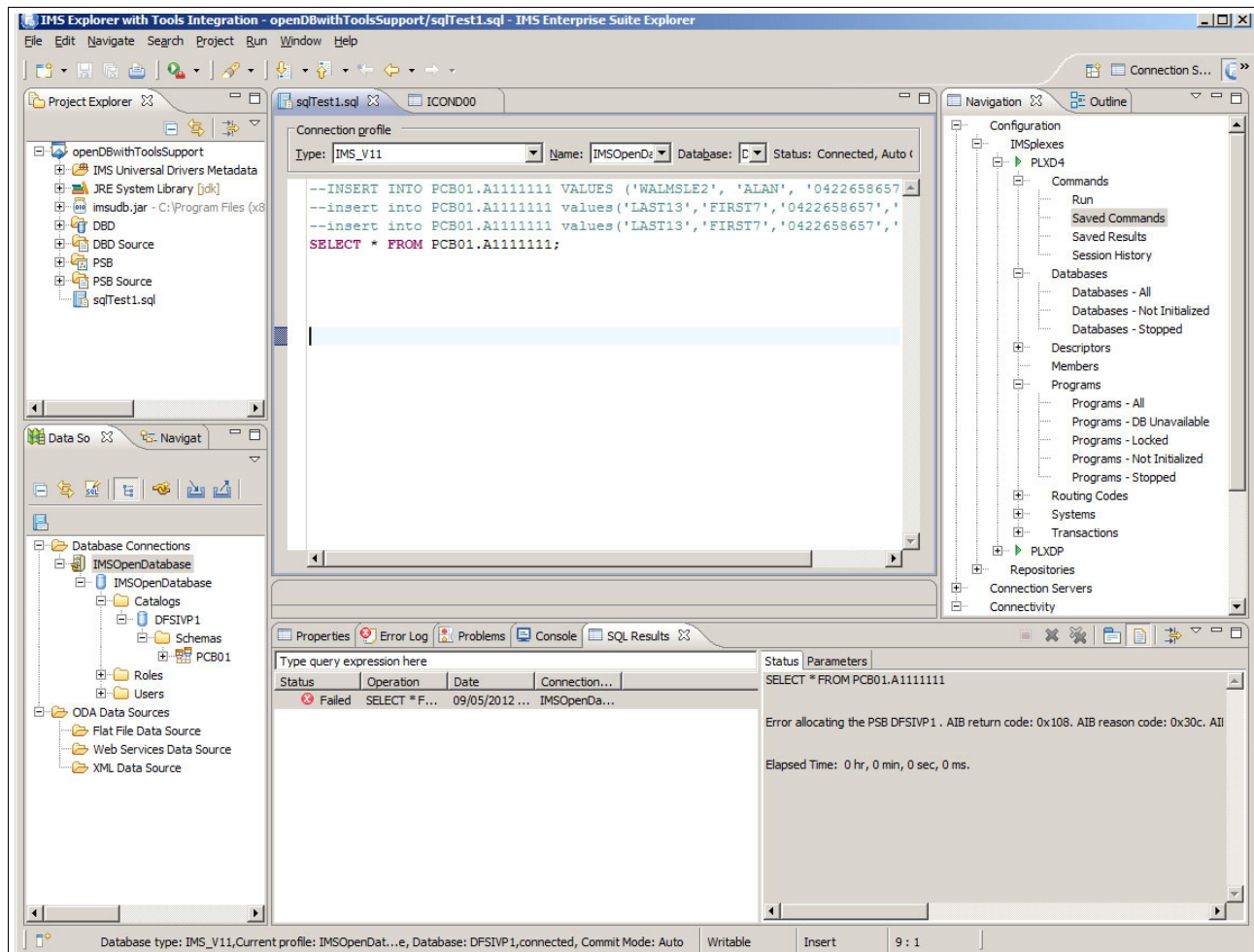


Figure 3-43 Diagnosing an SQL call failure

But, when running the application, the developer encounters an error. What steps can the developer take to understand the flow of the request and potentially even solve the problem by himself, without requiring system programmer intervention?

As shown in Figure 3-44, the developer can use the Active Sessions display from within the IMS Explorer. He is using highlighting to see the sessions that are generated by Open Database and get more information about them.

The screenshot shows the IMS Explorer application with the following components:

- Project Explorer (Left):** A tree view showing the project structure, including folders like 'openDBwithToolsSupport', 'IMS Universal Drivers Metadata', 'JRE System Library', 'imsudb.jar', 'DBD', 'DBD Source', 'PSB', 'PSB Source', and 'sqlTest1.sql'.
- Active Sessions Table (Center):** A table displaying session information. The columns are Session Type, Port, Socket, Event Key, and Start Time. Several rows are highlighted in green, indicating sessions generated by Open Database (ODBM).

Session Type	Port	Socket	Event Key	Start Time
OTMA	4101	5	CA6F2D826D4E7F06	2012-11-07 15.20.59.247847
ODBM	4105	6	CA6F384B2B73DD06	2012-11-07 16.09.14.096445
OTMA	4101	7	CA6F2D828DFD8B04	2012-11-07 15.20.59.381720
OTMA	4101	8	CA6F2D8299493C04	2012-11-07 15.20.59.427987
OTMA	4101	9	CA6F2D82A2DFAF04	2012-11-07 15.20.59.467258
ODBM	4105	10	CA6F384B2B73DD06	2012-11-07 16.09.14.096603
OTMA	4101	13	CA6F2D82D1DBA006	2012-11-07 15.20.59.659706
OTMA	4101	15	CA6F2D82E7385984	2012-11-07 15.20.59.747205
OTMA	4101	17	CA6F2D830398F586	2012-11-07 15.20.59.863439
OTMA	4101	18	CA6F2D830FFD4C84	2012-11-07 15.20.59.914196
OTMA	4101	19	CA6F2D831C692A84	2012-11-07 15.20.59.965074
OTMA	4101	21	CA6F2D833AA3E906	2012-11-07 15.21.00.088894
- Configuration Pane (Right):** A tree view showing the system configuration, including folders like 'Configuration', 'Connection Servers', 'Connectivity', 'IMS Connect Systems', 'All systems', 'Demo Systems', 'Connect Comma', 'IMS Commands', 'Message Log', 'Sessions', 'Status Monitor', 'HWSOPGS1 : ICON', 'HWSOPGS2 : ICON', 'HWSOPGS3 : ICON', 'MSC group', and 'OTMA / DB Group'.
- SQL Results Pane (Bottom):** A table displaying query results. The columns are Status, Operation, and Connection Profile. The results show a mix of 'Succeeded' and 'Failed' status for the query 'SELECT * FROM PCB01.A1111111'.

Status	Operation	Connection Profile
✓ Succeeded	SELECT * FROM PCB01.A1111111...	IMSOOpenDatabase
✗ Failed	SELECT * FROM PCB01.A1111111...	IMSOOpenDatabase
✗ Failed	SELECT * FROM PCB01.A1111111...	IMSOOpenDatabase
✓ Succeeded	SELECT * FROM PCB01.A1111111...	IMSOOpenDatabase
✗ Failed	SELECT * FROM PCB01.A1111111...	IMSOOpenDatabase
✓ Succeeded	SELECT * FROM PCB01.A1111111...	IMSOOpenDatabase
✓ Succeeded	SELECT * FROM PCB01.A1111111...	IMSOOpenDatabase
✗ Failed	SELECT * FROM PCB01.A1111111...	IMSOOpenDatabase
✗ Failed	SELECT * FROM PCB01.A1111111...	IMSOOpenDatabase

Figure 3-44 Open database (ODBM) sessions

Figure 3-45 shows that developers can also activate tracing and switch journals to create a log file. This gives more information that they can pass to their IMS system programmers in order for them to help solve the problem, or possibly fix the problem without the intervention of the system programmer.

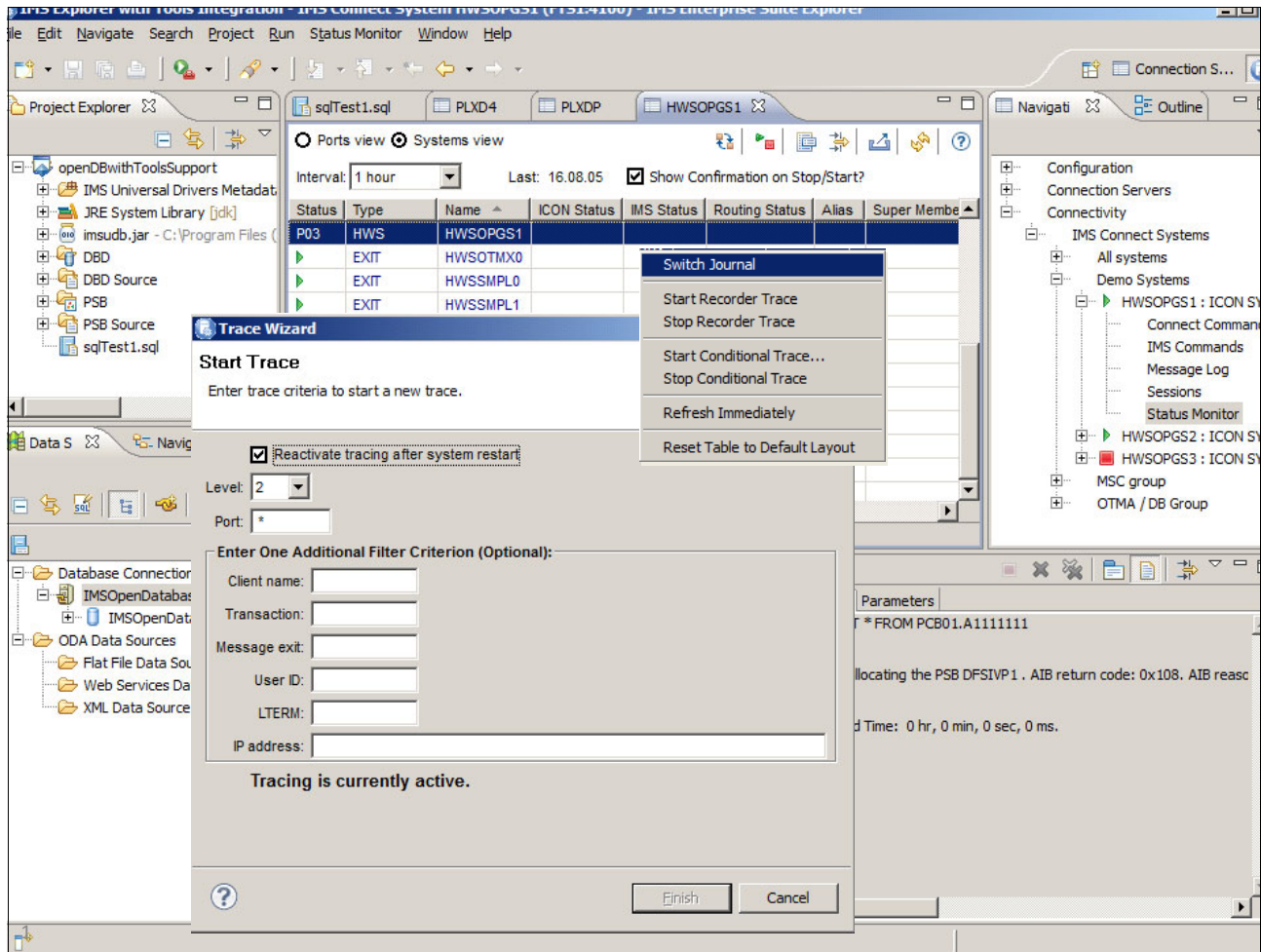


Figure 3-45 Initiating a trace

On the IMS side, IMS Connect Extensions collected logs of this transaction. In Figure 3-46, you can see the log data from this interaction.

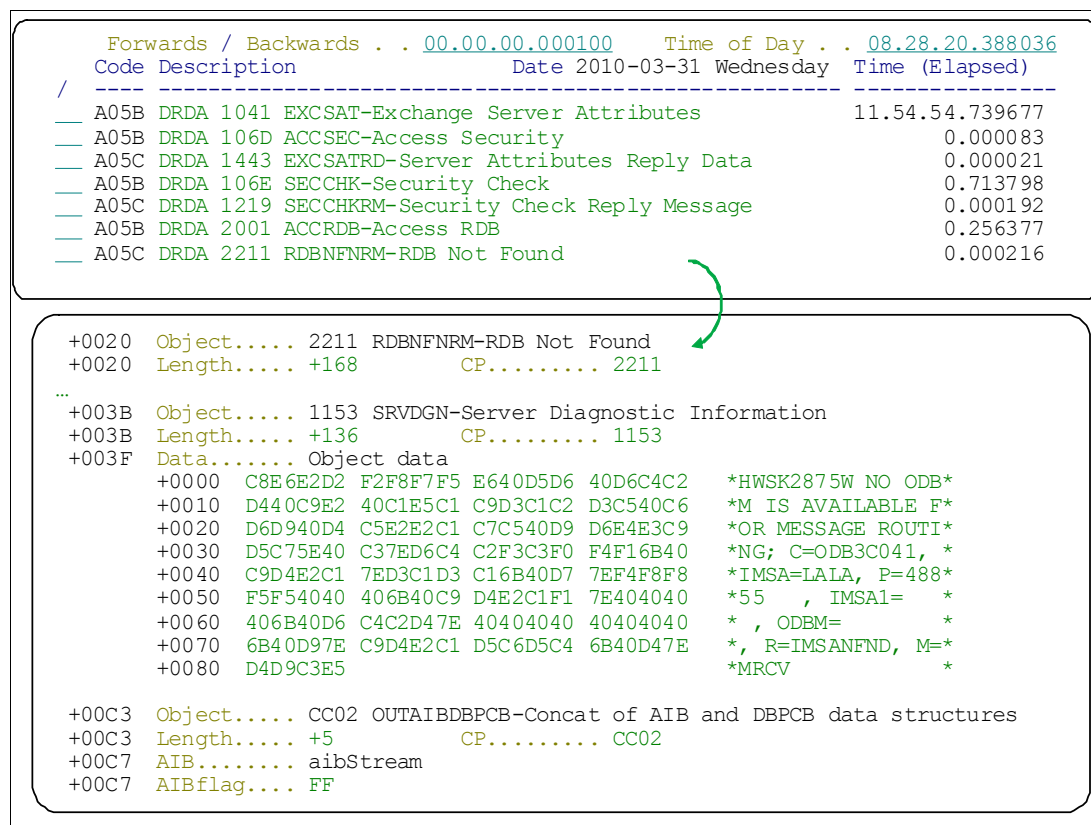


Figure 3-46 Browsing the journal records for an Open Database request



Open Transaction Manager Access

Open Transaction Manager Access (OTMA) is a function of IMS that was introduced with IMS Version 5. OTMA is a transaction-based, connectionless client/server protocol that provides an access path and an interface for sending and receiving IMS transactions and data.

OTMA is implemented for IMS in an z/OS Sysplex-capable environment and runs in the IMS control region. Instead of using VTAM or TCP/IP, OTMA uses the MVS Cross Coupling Facility (XCF), which provides functions to support cooperation between authorized programs running within a Sysplex. OTMA clients do not have to be in the same LPAR (z/OS image) as OTMA, but must be in the same Sysplex.

OTMA is a high-performance protocol that allows z/OS programs to access IMS applications. Because the OTMA connections use an internal cross-system communication that is almost comparable to main memory processing, OTMA has high performance characteristics.

This chapter covers the following topics:

- ▶ OTMA clients
- ▶ OTMA activation, operations, and command support
- ▶ OTMA activation, operations, and command support enhancements
- ▶ OTMA programming interfaces
- ▶ OTMA programming enhancements
- ▶ OTMA security
- ▶ OTMA security enhancements
- ▶ OTMA exit and descriptor usage
- ▶ OTMA exit and descriptor enhancements
- ▶ OTMA scalability and performance
- ▶ OTMA scalability and performance enhancements
- ▶ Diagnostic tests that are related to OTMA
- ▶ OTMA diagnostic enhancements

4.1 OTMA clients

There are a wide variety of OTMA clients. Here are the major ones.

4.1.1 IMS Connect is an OTMA Client for TCP/IP

IMS Connect is an IMS Open Transaction Manager Access client. Interactions from clients are passed through IMS Connect through TCP/IP connections. For more information about IMS Connect, see Chapter 2, “IMS Connect” on page 7.

4.1.2 IBM WebSphere MQ on z/OS as an IMS OTMA Client

There are two methods that WebSphere MQ can use to access the IMS environment:

- ▶ WebSphere MQ API calls:

The IMS application uses the WebSphere MQ API calls to get and put messages, including synchronization point coordination with IMS. Applications are bound with specific WebSphere MQ provided stub modules. For online programs, this requires connecting WebSphere MQ to IMS through an external subsystem (ESS) connection, similar to the IMS to DB2 subsystem connection. For batch, there is no ESS interface, and thus the sync point coordination requires z/OS RRS.

- ▶ As an OTMA client using the WebSphere MQ IMS bridge:

The WebSphere MQ IMS bridge is code in the WebSphere MQ queue manager, and it does not require connecting WebSphere MQ to IMS through ESS. The WebSphere MQ IMS bridge is an OTMA client that ships with WebSphere MQ queue manager code. The IMS bridge communicates with IMS using specially defined queues for taking the messages off the queue and sending them to IMS using the IMS OTMA interface, and receiving the output messages through that interface.

When the message arrives in WebSphere MQ, it is sent through XCF to the IMS OTMA interface. The message can be an IMS transaction or command, but it cannot be associated with an IMS LTERM. IMS puts it on the IMS message queue, and the application does a get unique (GU) call to the IOPCB to retrieve the message. This is similar to the implicit LU6.2 process. A remote queue manager can send a message to a local queue destined for IMS through OTMA.

The connection is defined in the WebSphere MQ CSQZPARM using the **OTMACON** keyword on the CSQ6SYSP macro. **OTMACON** has five positional parameters that are specified as follows:

`OTMACON = (Group,Member,Druexit,Age,Tpipepfx)`

The parameters have the following meanings:

Group	This is the name of the XCF group to which this instance of WebSphere MQ belongs.
Member	This is the member name of this instance of WebSphere MQ within the XCF group.
Druexit	The IMS destination resolution user exit that is used to format OTMA user data. The default name is DFSYDRU0. If overrides the OTMA Client Descriptor DFSYDTx. Consider using a name of DRU0xxxx (where xxxx is a WebSphere MQ queue manager name) for operational identification simplicity.

Age	How long (in seconds) a user ID from WebSphere MQ is considered previously verified in IMS.
Tpipepfx	A three-character prefix for a Tpipe name to avoid collision with IMS transaction code names. This is used each time WebSphere MQ creates a Tpipe; the rest of the name is assigned by WebSphere MQ. You cannot set the full Tpipe name for any Tpipe that is created by WebSphere MQ.

For WebSphere MQ, you must define one or more storage classes with the **XCFGNAME** and **XCFMNAME** parameters of the IMS systems to which you connect.

After it starts, WebSphere MQ joins the XCF group that is defined in the **OTMACON** parameter. The IMS bridge initiates a client bid resync to each active IMS that is defined in the **STGCLASS** macros. When the bid is successful, the IMS bridge opens the bridge queues and messages flow.

There are no WebSphere MQ commands to start or stop the IMS bridge although there are IMS commands:

- ▶ `/START OTMA - /STOP OTMA`
- ▶ `/START TMEMBER xxxx Tpipe yyyy - /STOP ...`

4.1.3 DB2 stored procedures include an OTMA Client: DSNAIMS

DSNAIMS is a stored procedure that enables DB2 applications to start IMS transactions and commands easily, without having to maintain their own connections to IMS.

This stored procedure uses the IMS OTMA Callable Interface (C/I) to connect to IMS and run the transactions. Figure 4-1 shows the general structure of DSNAIMS stored procedures. When you implement the DSNAIMS stored procedure to your DB2 UDB for z/OS environment, you can access an IMS transaction or IMS command by using the SQL call interface.

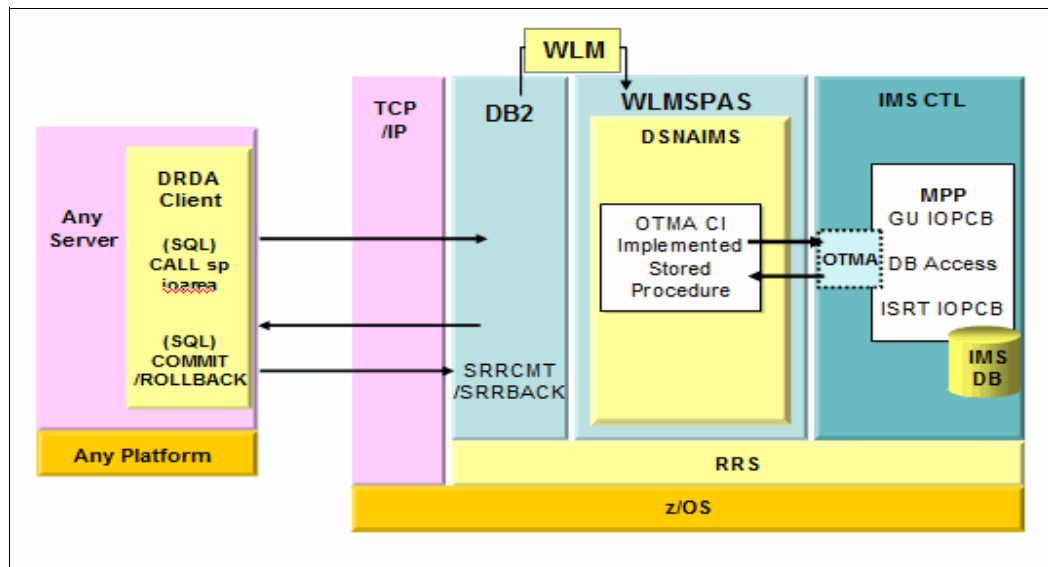


Figure 4-1 DSNAIMS stored procedure for OTMA C/I access

4.1.4 OTMA callable interface

You can also write your own OTMA Client by using the XCF interface or the OTMA Callable Interface, which is introduced with IMS Version 6.1, from Java, C, or C++ programs.

The IMS OTMA Callable Interface provides a high-level interface that allows application programs on other z/OS subsystems to access IMS applications through OTMA. The interface consists of API calls that can be used to join the IMS/OTMA XCF group, connect to IMS, allocate communication sessions, send IMS transactions and commands, receive output from IMS, close communication sessions, and leave an XCF group.

Figure 4-2 illustrates the OTMA environment and its interfaces.

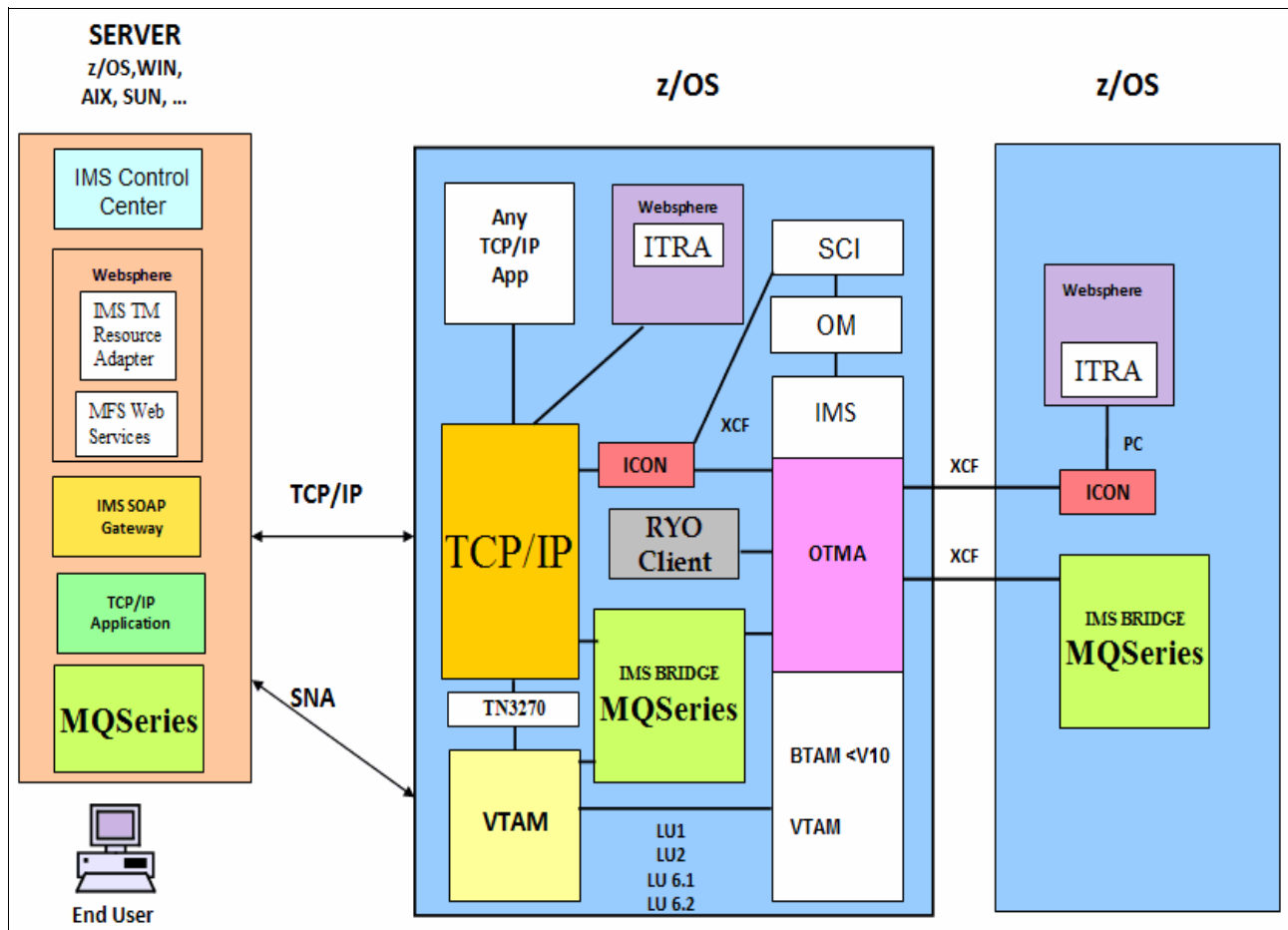


Figure 4-2 OTMA and its interfaces

4.2 OTMA activation, operations, and command support

The remainder of this chapter explains the base OTMA functions and then explains enhancements that were introduced in IMS Version 10, 11, 12, and 13, even though IMS Version 10 went out of service as of 12 November 2012. This explanation provides a roadmap of what enhancements you have available, based on your IMS Version in use.

The location of the OTMA protocol is restricted to the domain of the z/OS Cross-System Coupling Facility (XCF) transport layer. Each partner in an XCF-based protocol must belong to the same XCF group and must have a member name.

To enable IMS to use OTMA, specify the XCF group name and IMS OTMA member name during system definition. To start OTMA, you can set the **OTMA=Y** startup parameter in the IMS control region procedure or, after an IMS restart, run **type-1 command /START OTMA**.

All the following OTMA configuration parameters can be specified in the DFSPBxxx member of the IMS.PROCLIB data set and either of the IMS or DCC startup procedures.

- ▶ **GRNAME=**
Defines the name of the XCF group that IMS creates and joins for OTMA communications.
- ▶ **OTMA=**
Specifies whether OTMA starts during IMS start and whether the **/START OTMA** command is recoverable across warm and emergency restarts.
- ▶ **OTMAASY=**
For CM1 send-then commit messages, controls the synchronous or asynchronous scheduling of transaction originating from a program-to-program switch.
- ▶ **OTMANM=**
Specifies the XCF member name that identifies OTMA within the XCF group.
- ▶ **OTMAMD=**
Determines whether the member override field in the parameter list of the OTMA Destination Resolution user exit (OTMAYPRX) can be used to specify a different XCF member name for transactions that are invoked from an OTMA client.
- ▶ **OTMASP=**
For WebSphere MQ, specifies whether a synchronized Tpipe is used for OTMA output.
- ▶ **OTMASE=**
Specifies the type of OTMA RACF security that you want to use, if any.
Commands that are associated with OTMA operations are described in 2.10, “IMS Connect operations and command support” on page 42.

4.3 OTMA activation, operations, and command support enhancements

Most of IMS message processing options, such as non-response mode and response mode transactions, conversational processing, fast path transactions, Multiple Systems Coupling (MSC) processing, access through IMS Connect, and IMS commands can be implemented through OTMA. To support these various protocols, OTMA introduced many enhancements.

4.3.1 IMS Version 10: Message flood condition notification

You can define a limit to the number of input messages from any one OTMA client that can be waiting at the same time to be processed. A message flood condition occurs when too many transactions are waiting to be processed by OTMA. A message flood condition can use up all available LSQA storage and result in a z/OS S40D abend. This condition might be caused by either an abnormal increase in incoming messages or a problem that prevents IMS from processing the messages normally.

OTMA issues warning messages as the message count approaches the limit. When the message count reaches the limit, OTMA identifies a message flood condition and rejects any new messages from the OTMA clients until the message count drops or a new limit is set.

The message flood control enhancement in IMS Version 10 is enabled automatically with a default limit of 5000 messages. Here are the messages for the input message flood condition that are routed to the WTO system console:

- ▶ DFS1988W OTMA input messages from member yyyy have reached xx% of the maximum concurrent input message limit zzzz.
- ▶ DFS1989E OTMA input messages from member yyyy have reached the maximum concurrent input message limit zzzz.
- ▶ DFS0767I OTMA message flood condition was relieved for member yyyy.

To provide compatibility with previous releases and deactivate the support, either specify an INPUT value of zero in a descriptor or run `/STA TMEMBER INPUT 0`.

Value: Increased notification of flood conditions before storage or performance issues arise.

4.3.2 IMS Version 10: OTMA processing during IMS restart

In addition to the existing OTMA values of Y and N, IMS Version 10 introduces the option of **OTMA=M** (manual). When IMS first initializes, the value of **OTMA=M** functions similarly to **OTMA=N** so that OTMA is not started. After IMS is up and running, the `/STA OTMA` command can be run, but becomes unrecoverable.

The setting of **OTMA=M** takes effect when IMS terminates either normally or abnormally and must be restarted. During restart processing, OTMA is not automatically restarted. This capability was introduced to prevent looping abend situations where IMS might have terminated as a result of OTMA error conditions.

Value: Improved operational control of IMS restarts after initial OTMA failures.

4.3.3 IMS Version 10: /DIS TMEMBER Tpipe command enhanced

The `/DISPLAY TMEMBER Tpipe` command displays information for both commit-then-send and send-then-commit messages, including the accumulated input message count on the queue (wraps at 66535) and the name of the Destination Resolution (DRU) exit routine.

Value: Knowing the accumulated input message count on a Tpipe helps determine input queue lengths.

4.3.4 IMS Version 10: CM1 timeout controls

A send-then-commit (CM1) timer was introduced, which addresses output reply messages that OTMA sends for synclevel=confirm or synclevel=syncpt processing. When IMS waits in “wait-syncpoint” or “wait-RRS” status, locks are held, and the dependent region is occupied. A default timeout value of 120 seconds for ACK/NAK not received is introduced, which protects the IMS environment from remote applications that do not respond in a timely fashion or network delays.

There are several methods to introduce the CM1 timeout control:

- ▶ Use the **T/0** parameter in the OTMA descriptor member DFSYDTx in IMS.PROCLIB. **T/0** defines the timeout value for OTMA send-then-commit response messages. The value that is specified can be 0 - 255 seconds. If the value is 0, OTMA deactivates the timeout function. If it is over 255, it is set to 120, which is the default.
- ▶ Use the new **TIMEOUT** specification in the **/START TMBER** command.
- ▶ Use the new specification in the OTMA member request either when the member joins the group using the client-bid protocol message or, at a lower level of granularity, whenever a CM1 (send-then-commit message) flows into OTMA.

To provide compatibility with previous releases and to deactivate the support, specify a value of zero in a descriptor or run **/STA TMBER TIMEOUT 0**.

Value: This enhancement controls a potential “wait-syncpoint” or “wait-RRS” condition for the IMS dependent regions that processed the OTMA transactions.

4.3.5 IMS Version 11: Timeout capability for commit-then-send messages

As described in 4.3.4, “IMS Version 10: CM1 timeout controls” on page 133, IMS Version 10 provided a timeout control option that is applicable to send-then-commit (CM1) message processing.

As a review, OTMA CM1 response messages that are associated with synclevel=confirm or synclevel=syncpt requests require an ACK/NAK from the OTMA client. Because of the possibility of a client programming error or a network failure or delay, the expected ACK/NAK might not be received by IMS, resulting in IMS holding up resources during the sync point process. To resolve this situation, the OTMA CM1 timeout support provided an overridable timeout default of 120 seconds.

The methods that are used to override the new CM0 timeout specification 120 second default is the same as the CM1 timeout, which means it applies to both CM0 and CM1 output reply messages.

A timeout condition always results in a DFS3494E with the timeout information being sent to the system console and MTO.

IMS Connect takes advantage of the OTMA CM0 timeout support and provides a new parameter that is called **CM0ATOQ**, as described in 2.5.5, “IMS Version 11: CM0 ACK timeout support” on page 34, which can be used to provide a timeout queue name. Output messages queued to a hold queue can later be retrieved by running a Resume Tpipe command from a client.

Value: This enhancement addresses problems in the network or with programming errors on the remote side that result in a non-existent or lost ACK for the CM0 reply.

4.3.6 IMS Version 11: Enhanced resource monitoring

IMS Version 11 provides a new client/server protocol that allows OTMA to monitor certain resources that are used by OTMA functions so that an automatic detection of possible degraded levels of these resources can occur. If needed, OTMA can process appropriate server actions depending on the condition and send out a protocol message to the OTMA member client.

The OTMA member that receives the protocol messages can choose to take corrective action when a degraded condition occurs. One example is the rerouting of a transaction request from one IMS to another IMS if the target IMS is unable to accept the message. In the situation where none of the other IMS systems are suitable or available to accept a reroute, the OTMA client has the option of marking the IMS system “temporary unavailable” and then try again later”.

The new resource monitoring capability can therefore address the situation when an OTMA client (such as IMS Connect or WebSphere MQ) is unaware that a target IMS cannot process the workload and continues sending work to OTMA. This situation can lead to flooded transactions and an IMS outage, which disrupts all end client applications.

As covered in 4.3.1, “IMS Version 10: Message flood condition notification” on page 132, OTMA can automatically monitor the growth of active OTMA input messages and reject them when they reach a 100% threshold. In IMS Version 11, early warning detection of issues can occur through heartbeat monitoring and analysis of internal control blocks that are associated with messages as they are received from XCF from each OTMA member client. Potential elongated service times could be the result of the following items:

- ▶ RACF processing
- ▶ Queue manager processing
- ▶ Internal IMS problems

Additionally, OTMA can monitor the total number of active input messages for all OTMA members through the `/START TMBER ALL INPUT #####` command, where the **ALL** parameter provides a global value that OTMA additionally monitors for all combined members. OTMA uses the new specified limit instead of the system default of 8000 to monitor the total active input message count for send-then-commit (CM1) transactions from all of the OTMA members.

If the total number reaches this specified limit, OTMA issues message DFS4388W to the IMS MTO and system console along with OTMA protocol messages with the warning status to all of the OTMA clients. When the condition is relieved, OTMA issues a DFS0798I message to the IMS MTO and system console along with OTMA protocol messages reflecting a good status to all of the OTMA member clients. Additionally, because these are unsolicited messages, they are included in the Operations Manager (OM) audit trail if it is defined in your environment.

Value: Allows OTMA member clients to take advantage of early flood detection and failure notifications.

4.3.7 IMS Version 11: OTMA type-2 commands

Several new type-2 commands provide greater control over the OTMA environment. Through these enhancements, requests from a TSO SPOC or IMS Control Center client can display information that is relative to the OTMA workload, and request dynamic changes to the OTMA descriptors.

► **QUERY OTMATI**

The **QUERY OTMATI** commands monitor the workload in IMS OTMA, specifically the “transaction instances” or messages in the queue that are received and sent through OTMA (internally represented by a transaction instance block (TIB)). When the command is issued without parameters, for example, **QUERY OTMATI**, the information displays the TMEMLBER, the Tpipe under the TMEMLBER, and the total number of transaction instance control blocks, which may represent queued messages, continuing IMS conversations, or other problems.

► Four commands that are related to OTMA descriptors

The OTMA destination routing descriptors were introduced in IMS Version 10 as enhancements to the DFSYDTx PROCLIB member data set. Changes to the descriptors, however, required a scheduled outage or restart of IMS to take effect. With the new type-2 commands (**CREATE OTMADESC**, **UPDATE OTMADESC**, **DELETE OTMADESC**, and **QUERY OTMADESC**), changes to these destination routing descriptors can be issued dynamically without any interruption to a running IMS instance.

Value: There is a dynamic capability to request information and monitor the OTMA transaction instances and modify the OTMA destination routing descriptors.

4.3.8 IMS Version 13: /DISPLAY TMEMLBER Tpipe command enhanced

OTMA usability was improved by adding support for a generic character or wildcard at the end of the Tpipe name in the IMS command **/DISPLAY TMEMLBER Tpipe**. By using the wildcard character, a series of Tpipe queues that are similarly named can be easily displayed. The generic form of the Tpipe name can also be used with the **QCNT** or **SYNC** keywords. The Tpipe name with a generic character cannot be mixed with other Tpipe names.

Value: Generic character or wildcard support in the **/DIS TMEMLBER Tpipe** command allows for easier search for specific resources. The generic form of the Tpipe name can also be used with the **QCNT** or **SYNC** keywords.

4.3.9 IMS Version 13: OTMA global flood control enhancement

OTMA at the IMS Version 13 level provides a new option to enforce the global flood limit to reject new requests and protect IMS from storage exhaustion. Previous releases protected flood conditions for individual OTMA clients (TMEMLBERs) but provided only warning messages when the global flood limit of active OTMA input messages and the control blocks associated with these input requests was reached. Now, there is the ability to address the global flood condition rather than just be warned.

Specifically, when an OTMA client sends a message to IMS, OTMA internally creates a control block that is called the transaction instance block (TIB) to track each active input message. Each TIB requires approximately 8 KB of extended private area storage. Additionally, z/OS creates a contents directory entry (CDE) to map each TIB. Each CDE requires approximately 150 bytes of the local system queue area (LSQA) storage.

Normally, TIBs and their associated z/OS CDEs are freed and the storage is released by IMS, either after IMS enqueues the commit mode 0 (CM0) input message or after IMS returns a CM1 output message.

If several thousand OTMA input transactions are received from an OTMA member and are waiting to be processed, thousands of control blocks representing those requests take large amounts of IMS storage, which can affect the overall IMS operations in the system. To prevent this type of OTMA message flood condition from an OTMA member, OTMA stops receiving the input transactions from this member based on a maximum value for the number of TIBs allowed for that OTMA member in the system.

You can view information about the TIBs and the OTMA messages that are being processed by IMS by issuing the type-2 IMS command **QUERY OTMATI**.

However, suppressing the input OTMA transactions targets only individual OTMA members that have the maximum value or threshold of TIBs that are defined. This type of OTMA flood control does not protect the IMS system when it has multiple OTMA members. In this case, each one of them might not reach the flood limit for that member. However, the total TIBs from all the OTMA members can exceed the global flood limit of the IMS system. When this happens, the pre-IMS Versions 13 system issues a warning signal through the Global Flood Control capability that was introduced in IMS Version 11 to the system console, the MTO, and all the OTMA members, without suppressing the input OTMA transactions. New transactions, however, can still come into IMS and flood the system. This could cause the IMS system to fail with a S878 abend. Figure 4-3 provides an introduction to this enhancement.

- Global flood control enhancements
 - New defaults:
 - OTMA global flood limit: 8,000 → 10,000
 - Relief value: 6,400 → 5,000
 - New DFS3428W warning message at 80% and 5% increments until relief
 - Enhanced DFS4388W is issued when the limit is reached and the new support to reject input is not active
 - Actions when global flood control has not been requested/activated
 - Warning messages
 - Enhancements to request global flood control (0-99999 new max)
 - New meaning: /STA TMEBER ALL INPUT ###
 - New "global" OTMA Client descriptor: M DFSOTMA INPT=

Figure 4-3 Global flood control enhancement

The default OTMA global flood limit of the system was changed from 8,000 to 10,000, along with a default relief level change from 6400 to 5000. A new DFS3428W warning message was introduced that is sent when 80% of the global flood limit is reached. This message is issued every 5% increment thereafter until the global flood limit is reached. After the global flood limit is reached, the enhanced warning message DFS4388W is sent to the system console and MTO along with OTMA protocol messages reflecting a warning status to all the OTMA members *when the global flood control has not been activated*. This global flood status can be relieved when the input messages in the system are processed and the number of TIBs are reduced to 50% or less of the global limit. The enhanced relief message DFS0793I then is sent to the IMS MTO and system console along with OTMA protocol messages reflecting a good status to all the OTMA member clients.

The default actions take place if IMS is activated without limits to request input rejection with global flood control. This means that an IMS system without any specification or changes comes with the default OTMA global flood monitoring and a default global limit of 10,000. IMS sends out warning messages DFS3428W and DFS4388W to the system console and MTO along with protocol warning messages to all the OTMA members. No input OTMA transactions are suppressed. This default mode of OTMA global flood monitoring can be changed to suppress input transactions from all the OTMA members after a global flood control and limits are specified.

IMS Version 13 allows the activation of “global flood control” through two methods:

- ▶ Running **/STA TMEBER ALL INPUT #####**, where ##### is the global flood limit as in previous releases, but now causes a rejection of new input when the value is reached.
- ▶ Specifying a global limit value in a new OTMA client descriptor (DFSOTMA).

In IMS Version 13, the meaning and actions that are associated with the **/STA TMEBER ALL INPUT #####** command have changed. In IMS Version 11, when this command was issued, OTMA used the specified limit instead of the system default of 8000 to monitor the total active input message count for send-then-commit (CM1) transactions from all of the OTMA members. If the total number reached this specified limit, OTMA issued a DFS4388W to the IMS MTO and system console along with OTMA protocol messages with the warning status to all of the OTMA clients. In IMS Version 13, the limit is used for more than a warning and allows OTMA to prevent any new message from being accepted until the flood situation is relieved.

The **/DISPLAY OTMA** command of a degraded system shows “SERVER+FLOOD” in the user status of the OTMA server member when the global flood control is activated and the global flood limit is reached. Figure 4-4 describes the **/STA TMEBER ALL INPUT #####** command.

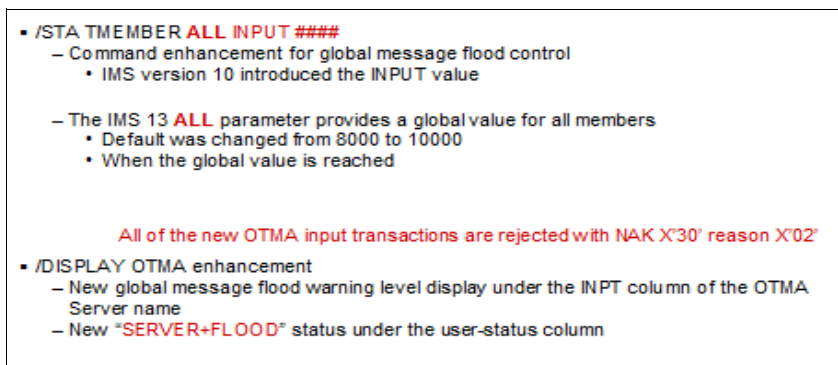


Figure 4-4 Commands and status that is related to GFC

The valid global flood limit has a new maximum value and can be specified as 0 - 99999. The valid range for the member flood limit continues to be 0 - 9999. As in previous releases, a non-zero value activates the global flood control for OTMA for the system. A value of 0 deactivates the global flood support and a value of 1 - 200 is treated as though 200 was specified.

The option also exists to use a descriptor to define the global flood control limit using a client descriptor. A special system client name, DFSOTMA, was introduced for this purpose. Activation of this DFSOTMA client descriptor enables the global flood control and requests suppression of any new OTMA transactions when the limit is reached. The parameter that applies to this function in the DFSOTMA client descriptor is **INPT=**. Specification of other parameters, such as **DRU=** and **T/O=** are ignored.

Figure 4-5 presents this concept.

- A special new client descriptor (optional)
 - Using existing 'M' descriptor type in DFSYDTx member of IMS.PROCLIB.

M client-name keywords

Where:

client-name is **DFSOTMA**

keywords are:

INPT= <VALID>
DRU= <Ignored>
T/O= <Ignored>
MAXTP= < >

For example: **M DFSOTMA INPT=22222**

- New system client name for all the OTMA members, DFSOTMA, is introduced to set the global flood limit via INPT parameter.

Figure 4-5 OTMA Client Descriptor DFSOTMA that is used to set the GFC value

Global flood control has action points at 80% (warning), 100% (rejection of new messages), and 50% (relief) of the global flood limit. When global flood control is activated and the global flood limit is reached, OTMA rejects all the new input transactions from all the members with OTMA sense code x'0030' and reason code x'0002'. Any synchronous program switch requests from the internal OTMA member DFSYICAL are also rejected. The new DFS3429E error message is then sent to the system console and MTO and a protocol message with the command type set to X'3C' is sent to all the OTMA members with an "unavailable for work" status, unlike previous releases, which sent only a "warning" status. When OTMA members receive this status, they can choose to take corrective action.

One example of a corrective action is the rerouting of all the new transaction requests from one IMS system to another. After IMS reaches the 50% value of the global limit for unprocessed messages, the global flood status is relieved and IMS begins accepting new input messages. The enhanced relief message DFS0793I is then sent to the IMS MTO and system console along with OTMA protocol messages reflecting a good status to all the OTMA member clients.

The MEMBER flood limit (versus GLOBAL for all members combined) remains at 9999 as in previous releases. When a member reaches the member flood limit, new transactions from this member are rejected with OTMA sense code x'0030' and the new reason code x'0001'.

Value: OTMA can autonomically address the global flood condition rather than just be warned.

4.3.10 OTMA global flood control demonstration

In this section, we present a series of six test scenarios that are associated with modifications of the OTMA Client Descriptor and what to expect with the IMS Version 13 level of OTMA global flood controls for YTIB control blocks. A YTIB is associated with an input transaction to OTMA.

The purpose of these test scenarios is to test the following items:

- ▶ Specifying the following global limit values in the OTMA client descriptor for all the OTMA members:
 - INPT=(1-200). The default should be set to 200.
 - INPT=(0-99999).
 - INPT=(999999). Should fail. The maximum is 99999.
- ▶ Test whether the **/STA TMEMBER ALL INPUT 0** command does in fact disable the global flood limit monitoring.
- ▶ Test to validate the warning, global flood limit, and the relief messages that are generated when their various conditions are reached.

The system configuration includes the following items:

- ▶ One IMS Version 13 image
- ▶ Three IMS Connect images that are defined as super member SM01, which serves this single IMS
- ▶ Three sample online applications to provide input messages through IMS Connect to OTMA
- ▶ One COBOL application to perform an ICAL that is associated with a program switch

Scenario 1: Initial check of the member level availability status code

Here are the major stages of this scenario:

- ▶ Define the three IMS Connect configurations, as shown in Example 4-1.

Example 4-1 IMS Connect configurations: HWSCFG01 HWSCFG02 and HWSCFG03

HWSCFG01

```
HWS=(ID=HWS1,XIBAREA=100,RACF=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(9999,LOCAL),RACFID=GOFISHIN,TIMEOUT=5000,
IPV6=Y,SSLPORT=(1111),SLENVAR=HWSCFSSL,
EXIT=(HWSIMS00,HWSIMS01,HWSMPL0,HWSMPL1,HWSCSL00,HWSCSL01))
DATASTORE=(ID=IMS1,GROUP=XCFGRP1,MEMBER=HWS1,TMEMBER=IMS1,DRU=HWSYDRUO,
APPL=APPLID1),SMEMBER=SM01
IMSPLEX=(MEMBER=IMSPLEX1,TMEMBER=PLEX1)
```

HWSCFG02

```
HWS=(ID=HWS2,XIBAREA=100,RACF=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(9998,LOCAL),RACFID=GOFISHIN,TIMEOUT=5000,
IPV6=Y,SSLPORT=(1111),SLENVAR=HWSCFSSL,
EXIT=(HWSIMS00,HWSIMS01,HWSMPL0,HWSMPL1,HWSCSL00,HWSCSL01))
DATASTORE=(ID=IMS1,GROUP=XCFGRP1,MEMBER=HWS2,TMEMBER=IMS1,DRU=HWSYDRUO,
APPL=APPLID1),SMEMBER=SM01
IMSPLEX=(MEMBER=IMSPLEX1,TMEMBER=PLEX1)
```

HWSCFG03

```
HWS=(ID=HWS3,XIBAREA=100,RACF=N)
TCPIP=(HOSTNAME=TCPIP,PORTID=(9997,LOCAL),RACFID=GOFISHIN,TIMEOUT=5000,
IPV6=Y,SSLPORT=(1111),SLENVAR=HWSCFSSL,
EXIT=(HWSIMS00,HWSIMS01,HWSMPL0,HWSMPL1,HWSCSL00,HWSCSL01))
DATASTORE=(ID=IMS1,GROUP=XCFGPR1,MEMBER=HWS3,TMEMBER=IMS1,DRU=HWSYDRU0,
APPL=APPLID1),SMEMBER=SM01
IMSPLEX=(MEMBER=IMSPLEX1,TMEMBER=PLEX1)
```

- ▶ Earlier in the test, the job that is shown in Example 4-2 was run to set the INPT value to 3 in the OTMA descriptor DFSYDTC. If the Global Flood Limit is set less than 200, then the default setting is forced to 200.

Example 4-2 Setting the INPT value of 3 in the OTMA descriptor DFSYDTC

```
//DFSYDTC JOB ('A=W005'),BATCH,CLASS=K,MSGLEVEL=(1,1)
/*ROUTE PRINT THISCPU/TIMPER
/*
//STEP1 EXEC PGM=IEBUPDTE,PARM=NEW
//SYSUT1 DD UNIT=SYSDA,DISP=SHR,VOL=SER=USER01,
// DSN=USER.PRIVATE.PROCLIB
//SYSUT2 DD UNIT=SYSDA,DISP=SHR,VOL=SER=USER01,
// DSN=USER.PRIVATE.PROCLIB
//SYSPRINT DD SYSOUT=A
//SYSIN DD DATA,DLM=ZZ
./ ADD NAME=DFSYDTC,LIST=ALL
D DEST0001 TYPE=IMSTRAN
M DFSOTMA INPT=3
ZZ
/*
//
```

- ▶ A **/DIS OTMA** command verifies that the INPT value is 200 for all members
- ▶ A **/START TMEMBER ALL INPUT 300** command increases the Global Flood Control (GFC) value.
- ▶ A **/DIS OTMA** command verifies that the INPT value is 300 for all of the three IMS Connects.
- ▶ Set all of the IMS Connect members back to the GFC value of 200. We use this GFC during subsequent scenarios.
- ▶ Specify in the OTMA Client Descriptor an invalid global flood limit (GFL) of 999999 and determine whether the expected error message occurs. The parameters for this task are shown in Example 4-3.

Example 4-3 Modifying the DFSYDTC member with an invalid INPT value of 999999

```
./ ADD NAME=DFSYDTC,LIST=ALL
D DEST0001 TYPE=IMSTRAN
M DFSOTMA INPT=999999
ZZ
/*
```

Here is the response message:

```
DFS2385E SYNTAX ERROR FOR DESCRIPTOR = DFSOTMA INPT  MUST BE 1 TO 5 CHARACTERS  
LONG IMS1
```

Scenario 2: Inputting CM1 TRAns from three clients and checking the status code

Here are the stages of this scenario:

- Input CM1 TRAns from three TCP/IP clients and check the status code. Sixty transactions are sent to HWS1, 80 to HWS2, and 160 to HWS3. This results in 300 input transactions that are queued to OTMA. Because no dependent regions are started, they remain on the queue.
- Verify that warning message DFS3428W is issued at 80%, 85%, 90%, and 95% of the 300 input message capacity. Verify that error message DFS3429E is issued when the 100% capacity limit is reached. We can see this process in Example 4-4.

Example 4-4 OTMA flood indication and action messages

```
DFS3428W THE TOTAL OTMA INPUT MESSAGES(TIB) HAVE REACHED 80% OF THE GLOBAL LIMIT 300 IMS1  
DFS3428W THE TOTAL OTMA INPUT MESSAGES(TIB) HAVE REACHED 85% OF THE GLOBAL LIMIT 300 IMS1  
DFS3428W THE TOTAL OTMA INPUT MESSAGES(TIB) HAVE REACHED 90% OF THE GLOBAL LIMIT 300 IMS1  
DFS3428W THE TOTAL OTMA INPUT MESSAGES(TIB) HAVE REACHED 95% OF THE GLOBAL LIMIT 300 IMS1  
DFS3429E THE TOTAL OTMA INPUT MESSAGES(TIB) HAVE REACHED THE GLOBAL LIMIT 300 IMS1
```

- As shown in Example 4-5, run **/DIS OTMA** and verify the user-status **SERVER+FLOOD** condition.

Example 4-5 /DIS OTMA associated with SERVER+FLOOD status

```
GROUP/MEMBER XCF-STATUS USER-STATUS SECURITY TIB INPT SMEM  
DRUEXIT T/O TPCNT ACEEAGE MAXTP  
XCFGRP1  
-IMS1 ACTIVE SERVER+FLOOD NONE 300 300  
-IMS1 N/A 0  
-HWS1 ACTIVE ACCEPT TRAFFIC NONE 60 200  
-HWS1 HWSYDRUO 120 1 999999 0  
-HWS2 ACTIVE ACCEPT TRAFFIC NONE 80 200  
-HWS2 HWSYDRUO 120 1 999999 0  
-HWS3 ACTIVE ACCEPT TRAFFIC NONE 160 200  
-HWS3 HWSYDRUO 120 1 999999 0  
*13198/210332* IMS1
```

SERVER+FLOOD indicates that the global flood limit was specified in the client descriptor or in the **/START TMEMBER ALL INPUT** command, and IMS reached this OTMA global flood limit.

Also, when a **VIEWHWS** is issued for each IMS Connect, you observe STATE=SEVERE within the DATASTORE section of the command output. This indicates that the OTMA server is experiencing some severe resource issues.

Scenario 3: Rejection of input after GFL is reached

This scenario is associated with running two applications: One to run the COBOL application and the second to warehouse the program switch, which processes an ICAL and then sends back a response synchronously. Two MPRs are started for these two applications to run.

You verify that any messages that are sent when GFL is at 100% are rejected.

The source for COBOL application IAPMDI27 can be reviewed in Appendix A.1, “ICAL Synchronous Program Switch COBOL program” on page 430.

Because the message that is associated with this flow is rejected, it is never housed in the IMS message queues.

A **/DIS OTMA** command that is entered after these applications still indicates 300 input messages on the IMS1 queue and that the **SERVER+FLOOD** condition still exists.

Scenario 4: Disabling Global Flood Controls

Here are the stages of this scenario:

- ▶ Run **/START TMEMBER ALL INPUT 0** to disable GFL.
- ▶ Verify the message DFS0793I and check the status after GFL is relieved.

This command and response message are listed in Example 4-6.

Example 4-6 /START TMEMBER ALL INPUT 0

```
R 45,/START TMEMBER ALL INPUT 0
IEE600I REPLY TO 45 IS;/START TMEMBER ALL INPUT 0
DFS058I 21:04:56 START COMMAND COMPLETED   IMS1
DFS0793I THE FLOOD CONDITION OF TOTAL OTMA INPUT MESSAGES(TIB) WAS
RELIEVED IMS1
```

- ▶ A **/DIS OTMA** command indicates that there are no input messages that are queued to IMS1 now. A **VIEWHWS** returns **STATE=AVAIL**.
- ▶ Run **/START TMEMBER ALL INPUT 300** to reset the GFC.

Scenario 5: GFL status after /CHE FREEZE and IMS warm start

Here are the stages of this scenario:

- ▶ Run **/CHE FREEZE** on IMS1.
- ▶ Warm start IMS1. Check to see that the TIBs are freed after warm start. Because the associated CM1 (send then commit) messages are always treated as unrecoverable, they are discarded from the IMS message queues after an IMS **/CHE FREEZE** command is run.

Many **HWSD0252W** messages were created during the **/CHE FREEZE**. **HWSD0242W** states the following message:

IMS Connect received a message from a data store but was not able to send the response to the required TCP/IP client. This situation can happen when the client that made the request is no longer active.

- ▶ Run **/DIS OTMA** to verify that the GFL is still 300 after warm start.
- ▶ From TCP/IP clients, input CM1 transactions to IMS Connect 1 and IMS Connect 2.

Scenario 6: GFL status after IMS cancel and emergency restart

Here are the stages of this scenario:

- ▶ Shut down IMS1 with a cancel. This results in a system **ABEND 222**.
- ▶ Emergency restart IMS1.
- ▶ Run **/DIS OTMA** to check that TIBs are freed after emergency restart and that the GFL is set to 300 after running **/ERE**. The GFL setting should remain over normal and emergency restarts.

4.3.11 IMS Version 13: OTMA MAXTP enhancements

A Tpipe is a logical connection between IMS and the OTMA client, through which all input and output for a client or client ID is routed and queued. OTMA creates Tpipes dynamically at the request of the OTMA client. Each Tpipe has extended CSA storage and extended private storage that is associated with it. When there are too many Tpipes in the IMS system, it can be a storage concern. To limit the growth of OTMA Tpipe creation in the system, the optional parameter, **MAXTP**, in the OTMA client descriptor was introduced to set a Tpipe limit for an OTMA member. After this parameter is defined for an OTMA member, OTMA begins monitoring requests for Tpipe creation.

The base support for OTMA MAXTP was previously introduced with APARs to IMS Version 11 and IMS Version 12:

- ▶ IMS 11: APAR PM14510 (PTFs UK65904/UK65905)
- ▶ IMS 12: APAR PM33681 (PTFs UK75918/UK75919)

There was another Special Product Enhancement (SPE) set of APARs that provided enhancements to the base support for OTMA MAXTP:

- ▶ IMS 11: APAR PM71035 (PTF UK91360)
- ▶ IMS 12: APAR PM74457

All of these enhancements are part of the IMS Version 13 base.

MAXTP is a parameter in the OTMA client descriptor that defines the maximum number of OTMA Tpipes that can be created for that member. After the parameter is defined, OTMA begins monitoring the Tpipe creation requests. This capability is beneficial for limiting the unrestrained growth of Tpipes, which could result in a S40D or other storage abend.

Figure 4-6 describes the parameters and shows some examples.

OTMA MAXTP SPE – Base Support

- Background
 - MAXTP parameter in the OTMA client descriptor in DFYSDTx
 - Defines the maximum number of OTMA TPIPEs that can be created for an OTMA member in an IMS system
 - Enables monitoring of TPIPE creation requests for the specified member
 - Limits the growth of TPIPEs

M client-name keywords Where:
client-name is the OTMA client XCF member name
keywords are:
DRU=name
INPT=0-99999
T/O=0-255
MAXTP=200-99999

indicates the maximum number of TPIPEs for this member that can be created. if the value is between 1 and 200, it is treated as 200. A value of 0 is ignored.

Examples:

M HWS1 MAXTP=3000

M HWS2 MAXTP=1000

Figure 4-6 Setting up the MAXTP parameter in OTMA client descriptor in DFYSDTx

After the **MAXTP** parameter is defined for an OTMA member, the request for the Tpipe creation for the OTMA member is monitored. Figure 4-7 maps out the monitoring process.

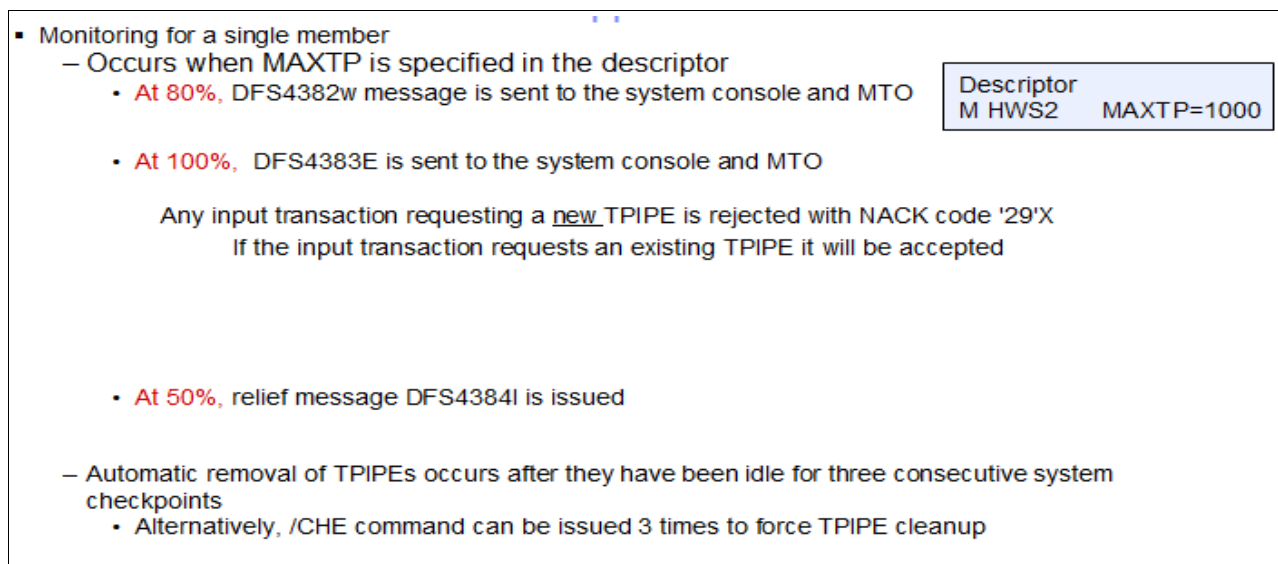


Figure 4-7 Monitoring actions and messages for a single OTMA member

When the total number of the Tpipes reaches 80% of the maximum, IMS generates a warning message DFS4382W to the system console and MTO. The DFS4382W message is displayed only once (in the situation, for example, where the percentage bounces from 80% to 75% to 84%, and so on). Only after the number drops down to 50% are the internal flags reset. After this reset, if the Tpipe number once again gets to 80%, then the DFS4382W message is issued.

When the total number of Tpipes gets to 100%, an error message DFS4383E is sent to the system console and MTO. Any input transaction requesting a new Tpipe is rejected with a new NACK code of x'29' when the maximum Tpipe limit is reached. If the input transaction requests an exiting Tpipe, it is accepted. Any asynchronous output that is rerouted with a new Tpipe name is ignored and the message is washed back to the original queue to be reprocessed. This is mainly for IMS Connect customers. Additionally, OTMA ALTPCB output messages specifying a new Tpipe name are not processed. The region receives an A1 or AX status.

When the number of Tpipes for the member drops back down to 50% of the MAXTP value, then relief message DFS4384I is issued.

IMS automatically attempts to remove transaction pipes after they are idle for three consecutive system checkpoints. You can issue /CHE commands three times to force the tpipe cleanup.

Figure 4-8 on page 145 shows the processing when the **MAXTP** values are being tracked for more than one member. When multiple OTMA member clients specify the **MAXTP** value in their client descriptors, the highest number that is defined among the members is treated as the global Tpipe limit for the members. When the total number of Tpipes in an IMS reaches the global limit, a warning message DFS4385W is sent to the system console and MTO. After the total number of the total Tpipes in an IMS drops down to 80% of the global limit, a relief message DFS4386I is sent out.

- Monitoring for all members that have MAXTP

- Highest number defined by any member becomes the global TPIPE limit
 - When the total number of TPIPEs in an IMS reaches the global limit

DFS4385W THE GLOBAL OTMA TPIPE COUNT 3000 HAS REACHED 100% OF 3000

- Once the total number of the total TPIPEs in an IMS drops down to 80%
 - Relief message DFS4386I is sent

DFS4386I THE GLOBAL OTMA TPIPE COUNT 2400 HAS DECREASED BELOW 80% OF 3000

Descriptors:

M HWS1	MAXTP=3000
M HWS2	MAXTP=1000

Figure 4-8 Monitoring for all members that have MAXTP

After the application of the enhancements is done, there still are a few issues. In a Shared Queues environment, U0367 abends occurred in back-end systems when the **MAXTP** condition was reached. Transactions that receive the abend are stopped and must be manually restarted. An enhancement that resolves this situation is described in 4.3.12, “IMS Version 13: MAXTP enhancement for the SQ environment” on page 145.

Additionally, the calculation for a global MAXTP value is based on picking the highest **MAXTP** value from the OTMA TMEMBERS that specify the value. The method precludes setting a value that accommodates even a partial accumulation of the individual specifications. An enhancement that is related to this limitation is described in 4.3.13, “IMS Version 13: Enhancements to MAXTP support” on page 146.

Value: This capability is beneficial for limiting the unrestrained growth of Tpipes, which can result in a S40D or other storage abend.

4.3.12 IMS Version 13: MAXTP enhancement for the SQ environment

This enhancement addresses shared queues back-end system abends. When an IMS shared queues back-end system attempted to process a front-end initiated OTMA non-fastpath input transaction at application GU time and the **MAXTP** was reached, the following situation occurred: Before IMS Version 13 or Version 11 with APAR PM71035 and Version 12 with APAR PM74457, customers using the OTMA MAXTP function in a shared queues environment experienced U0367 abends at the back-end IMS, and the transaction at the back end was stopped.

Now, the input message is discarded, the transaction is not stopped, and a DFS3323I message is sent to the front-end IMS to inform the OTMA client.

When a fastpath transaction is processed at the shared queues back-end IMS and experiences a **MAXTP** condition, the transaction is rejected and a DFS2193I message is sent back to the front-end IMS. The transaction in the shared queues back-end is not stopped.

Optionally, a new parameter **MAXTPBE** in the OTMA client descriptor of the shared queues back-end can be specified to tell IMS to honor [YES] or ignore [NO] the **MAXTP** checking at application GU time. Setting this parameter in the back-end IMS allows OTMA front-end initiated transactions to be scheduled without worrying about the number of Tpipe limit being defined.

Value: After the application of the SPE, when the situation described occurs, abend U0367 no longer occurs. Rather, the input message is discarded, the transaction is not stopped, and a DFS3323I message is sent to the front-end IMS to inform the OTMA client. This is a major operational and high availability enhancement.

4.3.13 IMS Version 13: Enhancements to MAXTP support

The enhancement for global **MAXTP** support leverages the DFSOTMA client descriptor and provides a **MAXTP** global limit for all the members. After this limit is specified, IMS no longer uses the highest limit that is defined by any of the members. The **MAXTP** limit in the DFSOTMA client descriptor activates a new way of monitoring the global Tpipe growth for all the members.

When 80% of this global limit is reached, a DFS4515W message is sent to the system console and MTO, and an OTMA protocol message reflecting a warning status is sent to all the OTMA members.

At 100%, when the global flood limit is reached, OTMA rejects all new Tpipe creation requests for all the OTMA members in the IMS. For an OTMA input transaction requesting a new Tpipe, a NAK rejection with OTMA sense code x'29' is sent. Additionally, a DFS4516E error message is sent to the system console and MTO, along with the OTMA protocol message reflecting a warning status to all the OTMA members. The reason why the protocol message shows the "warning" status instead of the "unavailable" status is that IMS can still accept transaction requests using existing Tpipes in the system. Running **/DISPLAY OTMA** on this degraded system also shows "MAX Tpipe" in the user status of the OTMA server member.

When the number of Tpipes for all the members is reduced to 50% of the user specified global limit, a relief message DFS4517I is sent to IMS system console and MTO. Additionally, an OTMA protocol message reflecting a good status is sent to all the OTMA members.

Value: Now you can easily set the **MAXTP** limit instead of defaulting to the highest limit that is defined in the members.

New MAXTPRL parameter

The new **MAXTPRL** parameter for the member descriptor sets the relief level of the Tpipe flood limit. The value can be 50 - 70. If the value is not specified, the default level is 50, which means 50% of the Tpipe **MAXTP** threshold flood limit. For value over 70, it is set to 70.

Value: This provides more control over the setting of the relief level percentage value.

Enhancements to the /DIS OTMA and /DIS TMEMBER

The **/DISPLAY OTMA** and **/DISPLAY TMEMBER** commands now show the **MAXTP** limits for the member and server in a new column. Previously, **MAXTP** for the member was not displayed. Additionally, the TPCNT column is enhanced to show existing Tpipe counts for members and server, even when the **MAXTP** parameter is not specified in the OTMA client descriptor. (Previously, TPCNT displayed Tpipe counts only when **MAXTP** was specified in the OTMA client descriptor.)

Value: Improved information is available from the `/DIS OTMA` and `/DISPLAY TMEBER` commands.

4.3.14 IMS Version 13: OTMA messages sent to both the MTO and WTO

There are many important IMS OTMA messages that are sent only to the Master Terminal Operator (MTO) LTERM. Many customers use automation for console Write To Operator (WTO) messages and can benefit from these OTMA messages being available. These messages are now being sent to both the MTO and WTO to the console. There are no changes to the content of the following messages:

- ▶ DFS0062W
- ▶ DFS1281E
- ▶ DFS1283E
- ▶ DFS1284E
- ▶ DFS1297E
- ▶ DFS2374W
- ▶ DFS2375W
- ▶ DFS2376W
- ▶ DFS2386I
- ▶ DFS2391I
- ▶ DFS2393I
- ▶ DFS2396I

Value: Allows access from the WTO for an important set of OTMA messages.

4.4 OTMA programming interfaces

The key to message flow for OTMA is the transaction pipe, which is the logical connection between the server and the OTMA client. An OTMA client includes the transaction-pipe name in the message-control information section of the message prefix for the input message. IMS then associates the application output for an OTMA client with a specific transaction pipe.

A transaction pipe is analogous to an IMS logical terminal (LTERM). For each LTERM, IMS maintains a connection between the queue and the physical node that receives the output. OTMA does not use an LTERM, but still must maintain a connection between the client and IMS. This connection is the transaction pipe, or Tpipe.

The OTMA client communicates with IMS by sending and receiving messages. It adds the OTMA message prefix to the input messages and it uses the message prefix to route the output data to the originating device or program. OTMA headers are written into the IMS log record type X'01' for OTMA input messages and into the IMS log record X'03' for responses to the OTMA destination. OTMA messages can be mapped by the DFSYMSG DSECT in SDFSMACT.

For more information about the message structure, see Chapter 47. "OTMA message prefix", in *IMS V13 Communications and Connections*, SC19-3651.

4.5 OTMA programming enhancements

The following IMS protocols are supported through OTMA:

- ▶ IMS conversations as unrecoverable send-then-commit (CM1) flows
- ▶ MSC processing OTMA transactions, including shared queues support
- ▶ Fast path transactions as send-then-commit (CM1)

Several important enhancements that are related to programming and operational support are listed in this section.

4.5.1 IMS Version 11: SQ ALTPCB back-end support

The original situation that this enhancement responded to was when an ALTPCB output message was generated on one of the back-end SQ systems and the remote client had no knowledge from which IMS the retrieval should be requested. Super member support initially allowed only retrieval of messages that were in the queue. It did not support creating a program that could automatically wait for new messages.

Now, OTMA super member support for Shared Queues provides the capability for an IMS Connect client to connect to any front-end IMS and retrieve an ALTPCB message that is created or will be created in any of the back-end systems in the Shared Queues group. This allows an IMS Connect Resume Tpipe request to retrieve an ALTPCB message from a Shared Queues back-end system.

Value: This enhancement allows a remote program to “listen” or “wait” for a message to be queued without concern as to which front-end IMS is used for the client connection. It simplifies remote program logic and enhances the generic routing and IP spraying support that IMS Connect provides for TCP/IP clients that access IMS environments.

4.5.2 IMS Version 11: Input message transaction expiration timeout

Transaction expiration support allows input messages whose time in IMS has exceeded an expiration value to be discarded before processing. The expiration value can be defined through system definitions through the **EXPRTIME** attribute in the **TRANSACTION** macro, through either the IMS DRD type-2 commands **CREATE TRANS | TRANDESC** and **UPDATE TRANS | TRANDESC**, or through the Output Creation Exit Routine **DFSINSX0**.

The OTMA environment supports two levels of specification of when an unprocessed input message should expire:

- ▶ Transaction level specification
- ▶ Enhanced support for message level specification. This allows an OTMA message to carry an expiration value using one of two methods:
 - An expiration Store Clock (STCK) time. This is supported by IMS Connect.
 - An elapsed time for transaction, similar to **EXPRTIME** value.

If the expired transaction condition is detected during the GU phase, an Abend U0243 is issued along with the DFS555I message unless the cancellation occurs in a Shared Queues back-end system, in which case a DFS2241I message is issued.

Value: OTMA environments have the flexibility of using the transaction level expiration support or message level overrides.

4.5.3 IMS Version 12: Message DFS2082I for CM0

The DFS2082I (response mode transactions terminated without reply) message is a mechanism to release an outstanding wait for CM1 (send-then-commit) transactions when the IMS application flow (across program-to-program switches) terminates without inserting a reply to the IOPCB. When converting remote programs that use CM1 to CM0 (commit-then-send) protocols, a problem can arise. Before IMS Version 12, a remote program that sends a CM0 transaction message then waits for a reply could wait a long time until a timeout occurs, if the IMS application is one that does not reply to the IOPCB.

IMS OTMA in Version 12 resolves this potential problem by introducing a new commit-then-send (CM0) optional flag to request the DFS2082 message. The flag **THAMHRSP** is set in the OTMA state data prefix. When this flag is specified for an input commit-then-send transaction, and the IMS application does not reply to the IOPCB or message switch to another transaction, OTMA sends a DFS2082 message to the client regardless of the IMS transaction response mode. This DFS2082 message for a commit-then-send transaction occurs only for the original input transaction and does not support the program-to-program switch. This restriction means that there are no DFS2082 messages for a switched to transaction, even if the switched to transaction fails to reply and the original transaction does not reply either.

Value: This eases CM1 to CM0 application conversions and reduces the unnecessary timeout in remote applications. This could impact the performance if the remote application must wait for a timeout before continuing to the next request.

4.5.4 IMS Version 13: Synchronous Callout SendOnly Ack

The Synchronous Callout SendOnly ACK Special Product Enhancement (SPE) addresses IMS Version 12 environments and is part of the base IMS Version 13.

This is a new call for remote servers to use when sending a reply to an IMS synchronous callout (ICAL) interaction. To enable the function in IMS Version 12, both APARs must be applied: one to IMS OTMA and the other to IMS Connect:

- ▶ IMS Connect Version 12: APAR PM39569 (PTF UK74666)
- ▶ IMS OTMA Version 12: APAR PM39562 (PTF UK74653)

This support allows a synchronous callout server, that is, the target of IMS's outbound ICAL, to process the request and, when sending the response, to request that IMS acknowledge receipt of the response. Figure 4-9 gives an example of the callout process, including a request for SendOnly ACK.

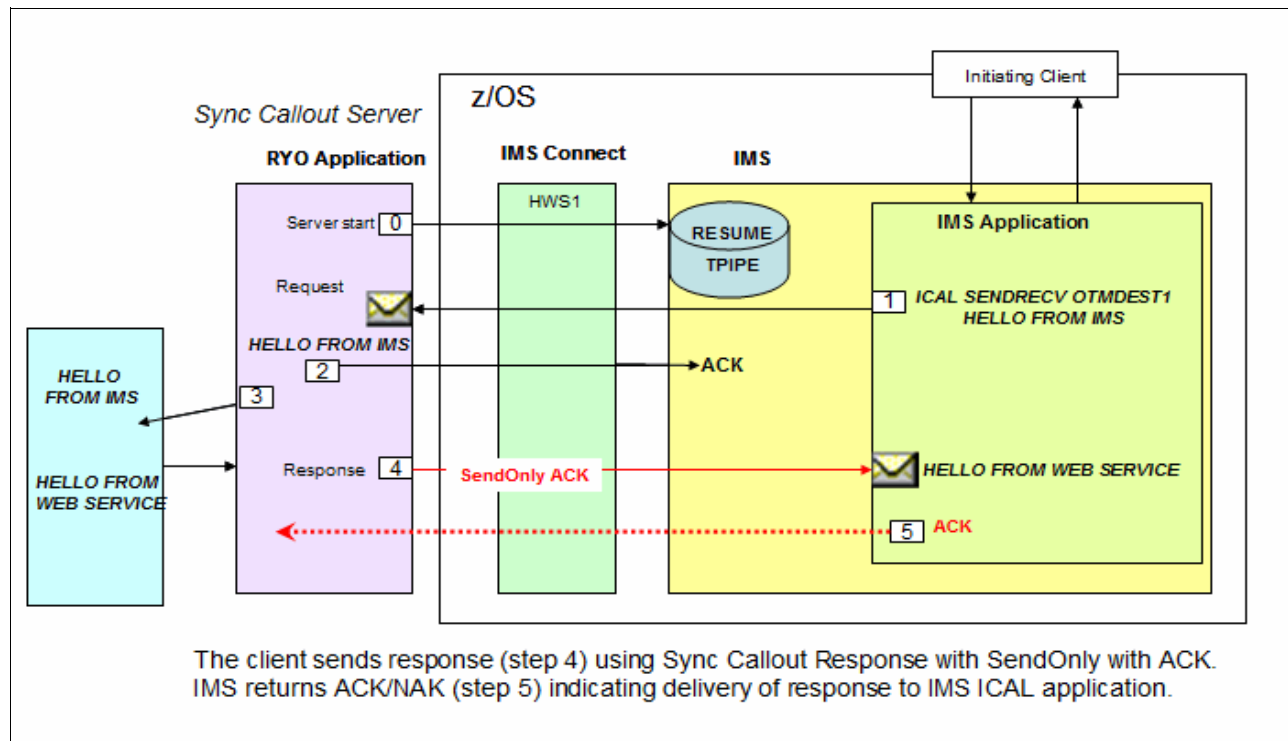


Figure 4-9 IMS 12 Synchronous Callout SendOnly Ack SPE

Value: Synchronous Callout servers can request and receive an indication of response delivery to the IMS ICAL application so they can commit or abort their updated resources.

4.5.5 IMS Version 13: OTMA CI enhanced async support

Before this enhancement, OTMA CI asynchronous messages (CM0) that are sent using `otma_send_async_API` have the responses sent immediately upon availability by OTMA because there is no HOLDQ. OTMA CI must “store” these messages in XCF until the client issues `otma_receive_async_API`. This has led to many timing issues between OTMA CI and the client.

This enhancement changes OTMA CI to use the OTMA HOLDQ. OTMA CI specifies it is HOLDQ capable on the client-bid. The `otma_send_async_API` was changed to use SendOnly with ACK protocol. The asynchronous output is placed on the HOLDQ. The `otma_receive_async_API` now uses Resume Tpipe Single to retrieve the output.

When a CM0 message is sent to an OTMA CI client, and is pending acknowledgement (ACK/NACK), the OTMA Tpipe status is now WAIT-H instead of WAIT-A.

Operational consideration

Because OTMA CI now uses the HOLDQ, the IMS Version 13 command **/DIS TMEBER xxx Tpipe yyy OUTPUT** can be used to display the Tpipe status. This command displays queue counts and status for both Primary and HOLDQ.

Migration consideration

OTMA CI clients no longer receive IMS DFS messages when OTMA sends a NAK for the input for asynchronous messages, but receive a return and reason code indicating the specific error.

Rather than receiving DFS064 for an invalid transaction code, the OTMA C/I client now receives RC=00000014, RSN=0000001A,0000001D,00000000,00000002.

- ▶ RC=x'14' indicates OTMA NAK codes.
- ▶ RSN=x'1A' indicates message cancelled
- ▶ RSN=x'1D' indicates SMB not found (invalid transaction code).

DFSYDRU0: OTMA User Data Formatting exit routine enhancements

Now, you can specify that OTMA CI uses the HOLDQ when asynchronous output is created before the OTMA CI client is established through client-bid. This is optional as any queued output is moved by OTMA to the HOLDQ after the OTMA CI client connects and specifies that it is HOLDQ capable.

Value: These enhancements eliminate timing and serialization issues when sending CMO output messages to the OTMA CI clients. There are no changes that are required for OTMA CI client applications.

4.6 OTMA security

OTMA provides various levels of security checking that can be implemented that is independent of the optional RACF verification that can be performed by IMS Connect. OTMA verifies security in two instances:

- ▶ When a client bids to connect
- ▶ When an input message from a client is processed

4.6.1 Client bids to connect

When a client sends a client bid command to OTMA to connect, OTMA must verify that the client is sufficiently authorized to connect to OTMA. The data that is passed in the security-data section of the client bid message is used by RACF to verify the client's authorization. The IMSXCF.xcf_group_name.xcf_member_name (client) must be defined to RACF in the FACILITY class. IMS Connect adheres to this protocol.

4.6.2 Processing an input message from the client

When OTMA processes an input message, whether it is command or transaction input, security is driven by the following items:

- ▶ Status that is set by the **OTMASE=** parameter in DFSPBxxx member
- ▶ Status that is set by the **IMS /SECURE OTMA** command
- ▶ Security data in the OTMA header

The **/SECURE** command controls the RACF security level for input from OTMA clients. It is used for administrative control of the IMS environment and as an emergency operations control command to throttle RACF activity, without requiring an IMS shutdown. The **/SEC OTMA** command is used with the **CHECK**, **FULL**, **NONE**, or **PROFILE** parameters. You can use the **/DISPLAY OTMA** command to show the security level that is in effect.

Use the **OTMASE** execution parameter to change the level of security at IMS start. The default is **FULL**. The **/SECURE OTMA** command overrides the value that you specify in the **OTMASE** keyword. If you do not specify the **OTMASE** keyword, IMS retains the OTMA security settings (which are established by the **/SECURE OTMA** command) after a warm start or emergency restart.

Here are the parameters of the **/SEC OTMA** command:

► **CHECK TMEMBER *tmembername***

Causes existing RACF calls to be made. IMS commands are checked by using the RACF resource class of CIMS. IMS transactions are checked by using TIMS.

► **FULL TMEMBER *tmembername***

Causes the same processing as the **CHECK** parameter, but uses additional RACF calls to create the security environment for dependent regions.

► **NONE TMEMBER *tmembername***

Does not call RACF within IMS for security verification.

► **PROFILE TMEMBER *tmembername***

Causes the values in the security-data section of the OTMA message prefix for each transaction to be used.

► **REFRESH TMEMBER *tmembername***

OTMA caches the ACEE for a user ID to reduce the amount of RACF I/O. As a result, a refresh for the cached ACEE is needed after the RACF database is updated. Running the **/SEC OTMA REFRESH** command without the **TMEMBER** option performs the ACEE refresh for all user IDs for all the OTMA clients. However, the ACEE refresh occurs when the next OTMA message for the user ID is received. This is designed to prevent all the RACF ACEE refreshes from happening at one time.

When **USER** is specified, OTMA refreshes, across all TMEMBERS, only ACEEs that include the specified user profile.

Note: The user exits, DFSCCMD0 and DFSCTRN0, if present, are started regardless of the status of the **/SECURE OTMA** command. This provides the facility to drive any user-written security from the appropriate exits.

4.6.3 Resuming transaction pipe security

Resume transaction pipe (Tpipe) support adds security to asynchronous output messages that a Resume Tpipe retrieves. Here are the support highlights:

- A new RIMS SAF/RACF security resource class:
 - Resume Tpipe Security is activated when RIMS is defined.
 - It associates the Tpipe name with the user ID/group that can access it.
 - Defines the Tpipe names as resources and PERMIT user IDs/groups.
 - You can specify RCLASS=Rxxxxxxx on the Security Macro or in DFSDCxxx.
- Requires an OTMA Security setting of **CHECK** or **FULL**.

- The OTMA Security User Exit Routine (DFSYRTUX) is used:
 - It is started after a call to SAF/RACF, regardless of the result.
 - It is always called, if it exists, whether RIMS is defined.
 - The default routine always provides RC=0 (allow access) for compatibility.

4.6.4 OTMA callable interface security

A RACF FACILITY class IMSXCF.OTMACI must be defined for the OTMA C/I to protect XCF groups from any non-authorized caller. When the RACF resource is defined, RACF **RACHECK** is started before OTMA C/I performs an XCF JOIN. This method protects the access to XCF, the XCF group, and the member. This RACF checking is performed only when a non-authorized caller is using OTMA C/I. Additional security characteristics remain consistent with OTMA.

4.6.5 IMS OTMA callout security

IMS transactions and commands that flow through OTMA from various clients are protected by current security classes, namely IMS and CIMS. The responses are ensured to be delivered to the client that initiated the transactions and commands. However, output messages in the hold queue that are generated as a result of asynchronous processing are not protected by any security class. When those messages are retrieved by an OTMA client using **RESUME Tpipe**, a security exposure can occur.

The function that is provided by callout security protects these output messages by establishing a security class named RIMS within RACF or any non-IBM security product. Within this class, the security definitions are associated with the Tpipe name and with the list of user IDs or group names under this Tpipe.

The enhancement allows IMS installations to optionally authorize the user ID, together with the Tpipe name that is contained in the **RESUME Tpipe** command message, to receive the output messages before any of these messages are sent to an OTMA client.

Implementation of callout security

IMS installations that want to take advantage of the callout security must define a new resource class, Tpipe name, and user IDs in RACF.

The resource class consists of the resource class type of “R” and the resource class name whose value is taken from the **RCLASS** parameter of the IMSCTRL macro. If **RCLASS** is omitted, the resource class name defaults to “IMS.” The resulting class then is “Rxxxxxxx”, where ‘xxxxxxx’ is the value of **RCLASS**, or “RIMS”, as the default.

Ensure that the OTMA client includes the following items:

- The Tpipe name in the OTMA CTL prefix
- The user ID in the security prefix header, in the message for the **RESUME Tpipe** command.

Code, assemble, and link edit the user exit in a library that is concatenated with the IMS RESLIB under DD name STEPLIB or JOBLIB. The DFSYRTUX user exit is always called regardless of the success or failure in RACF authorization.

Authorization is performed when **RESUME Tpipe** is initiated, but before retrieving the messages from the hold queue. The authorization procedure consists of two security levels:

- ▶ The first security level is to determine whether resource class “RIMS” or “Rxxxxxxx” is defined in the system. If it is not defined, the messages in the hold queue can be sent to the user through the OTMA client only after the user exit is started. If it is defined, the authorization logic not only verifies and validates the security header, but also authorizes the user ID under the Tpipe name using RACF facilities.
- ▶ The second security level is to start the user exit regardless of the success or failure of the first security level. The user exit can either take whatever results of the first security level, override its result, or add more restrictive security rules.

When authorization is successful, output messages in the hold queue are returned to IMS Connect. A rejection message of the **RESUME Tpipe** command is sent to the client when authorization fails.

How this function is started

The user either defines a new resource class in RACF or codes the user exit to supplement the authorization of the user ID under a Tpipe name. At minimum, the DFSYRTUX user exit is started even if the resource class, RIMS or Rxxxxxxx, is not defined in RACF. For more information about defining a resource class in RACF, see *Security Server RACF Security Administrator's Guide*, SA22-7683-14.

4.7 OTMA security enhancements

As the usage of OTMA has increased in many worldwide production environments, the requirements to secure access through it are vitally importance.

4.7.1 IMS Version 10: RESUME Tpipe security

IMS transactions and commands that flow through OTMA from various clients are protected by security classes, namely TIMS and CIMS. The responses are ensured to be delivered to the client that initiated the transactions and commands. Output messages in the hold queue that are generated as a result of asynchronous processing, however, are not protected by any security class. When those messages are retrieved by an OTMA client by running **RESUME Tpipe**, a security exposure can occur.

The function that is provided by **RESUME Tpipe** protects these output messages by establishing a security class named RIMS within RACF or any non-IBM security product. Within this class, the security definitions are associated with the Tpipe name along with the list of user IDs or group names under this Tpipe. The enhancement allows IMS installations to optionally authorize the user ID, together with the Tpipe name that is contained in the **RESUME Tpipe** command message before any of these messages are sent to a client.

The resource class consists of the resource class type of “R” and the resource class name whose value is taken from the **RCLASS** parameter of the SECURITY macro or in the DFSDCxxx member of proclib. If **RCLASS** is omitted, the resource class name defaults to “IMS.” The resulting class, then, is “Rxxxxxxx”, where xxxxxxxx is the value of **RCLASS** or “RIMS” as the default.

OTMA also provides the DFSYRTUX security exit routine as an opportunity to overrule the SAF/RACF decision or to extend the security check to allow modifications as needed by the environment.

Value: This enhancement addresses security in the destination routing environment.

4.7.2 IMS Version 10: OTMA instance-specific security levels

Before IMS Version 10, OTMA did not allow different security levels to be defined for various members. The security setting that is requested is considered a system-wide setting for all of OTMA members. In IMS Version 10, the OTMA command, **/SECURE OTMA**, is enhanced to allow specification of member security so that each OTMA client can have its own security level.

Use the **/SECURE OTMA *security-option* TMEMBER *member-name*** (where *security-option* is **FULL CHECK NONE PROFILE**) command to make member security specifications.

Value: You can specify security levels for each OTMA client independently, which provides much greater flexibility in tailoring OTMA security.

4.7.3 IMS Version 12: OTMA ACEE reduction for multiple OTMA clients

Multiple instances of OTMA member clients (TMEMBERS), for example, IMS Connect, DB2 Stored Procedures, and WebSphere MQ, are able to run in parallel on multiple servers or LPARs while still being able to connect to the same IMS. In previous releases, IMS OTMA isolated the security environment of one member client from another. This meant that an Access Control Environment Element (ACEE) for a user ID was created for each OTMA member client instance if the same user ID accessed IMS through more than one path, for example, IMS Connect and WebSphere MQ. Creating multiple copies of the same ACEE resulted in increased storage usage in subpool 249 in EPVT.

In IMS Version 12, OTMA caches the ACEE so that only one copy exists for the same user even when messages from that user are sent in through different member clients. The cached ACEEs also are in SP 249.

IMS Version 12 also introduces a new maximum aging value for ACEEs. The ACEE aging value triggers when an ACEE is refreshed. This value is specified by the OTMA member client during the client-bid process. The maximum was reduced from 68 years to 11.5 days. Each cached OTMA ACEE has an aging value that is based on the OTMA member client using it that has specified the lowest number.

Value: This enhancement is important because some OTMA clients, such as the DB2 Stored Procedure DSNAIMS, set the '7FFFFFFFF'X seconds (68 years) as the ACEE aging value for users sending IMS transactions and commands. It is now overridden to a maximum of 11.5 days. The enhancement also detects obsolete ACEEs, thus improving security.

4.8 OTMA exit and descriptor usage

There are several OTMA exits and two types of descriptors that are available for the customization of your OTMA environment. This section summarizes the four OTMA exits that are available for your usage.

4.8.1 OTMA Destination Resolution user exit (OTMAYPRX)

The OTMA Destination Resolution user exit determines whether an asynchronous output message must be routed to an OTMA destination or a non-OTMA destination. If the message should be routed to an OTMA destination, the user exit can determine the final OTMA destination client or Tpipe.

You can name this exit routine DFSYPRX0 and link it into a library that is included in the STEPLIB concatenation. Also, you can use OTMA destination descriptors as an alternative to coding an OTMAYPRX user exit.

4.8.2 OTMA Input/Output Edit user exit (OTMAIOED)

You can use the OTMA Input/Output Edit user exit to modify or cancel OTMA input and output messages. You can also use this user exit to format the User Prefix section of an OTMA input or output message.

You can name this user exit DFSYIOE0 and link it into a library that is included in the STEPLIB concatenation.

4.8.3 OTMA User Data Formatting exit routine (DFSYDRU0)

The DFSYDRU0 exit routine can change the final destination of OTMA messages by specifying OTMA member names, transaction pipe (Tpipe) names, or names of remote IMS systems. IMS Connect provides a sample OTMA User Data Formatting exit routine named HWSYDRU0.

4.8.4 OTMA Resume Tpipe Security user exit (OTMARTUX)

The OTMA Resume Tpipe Security user exit (OTMARTUX) provides a layer of security for **RESUME Tpipe** calls that are issued to retrieve messages queued to the OTMA asynchronous hold queue.

4.8.5 OTMA descriptors

The OTMA destination descriptors, which were introduced in IMS Version 10, can eliminate the need to code OTMA routing user exit routines by providing an easier way to specify the values and options that previously could be specified only by the exit routines. IMS stores OTMA destination descriptors in the DFSYDTx member of IMS procedure library (IMS PROCLIB). During start, IMS reads any predefined descriptors in the DFSYDTx member and initializes them.

After IMS start, you can use IMS type-2 commands to query, create, update, or delete the OTMA destination descriptors, which makes managing OTMA messages even easier.

OTMA provides two types of descriptors: an OTMA client descriptor, and an OTMA destination descriptor.

OTMA client descriptors

Use OTMA client descriptors to provide information about a specific OTMA client to IMS.

OTMA destination descriptors

OTMA destination descriptors provide a simpler method for describing destinations than using the OTMA Destination Resolution user exit (OTMAYPRX) and the OTMA User Data Formatting exit routine (DFSYDRU0).

OTMA destination types include the following ones:

► **TYPE=IMSCON**

OTMA destination descriptors support the routing of callout requests from IMS application programs to external data or service providers through IMS Connect. IMS Connect routes callout requests through one of the following IMS Connect clients:

- IMS Transaction Manager (TM) Resource Adapter
- IMS Enterprise Suite SOAP Gateway
- User-written IMS Connect clients

► **TYPE=IMSTRAN**

This type is used for other IMS application programs through synchronous program switch. You can route a message directly to another IMS application program by creating a descriptor with TYPE=IMSTRAN.

► **TYPE=MQSERIES**

OTMA can route asynchronous callout requests to WebSphere MQ with a TYPE=MQSERIES descriptor.

► **TYPE=NONOTMA**

If you are routing your messages to a destination such as an SNA terminal or a printer, define your destination type as non-OTMA (TYPE=NONOTMA).

4.9 OTMA exit and descriptor enhancements

Both IMS Version 10 and Version 13 introduced enhancements to OTMA exit and descriptor support.

4.9.1 IMS Version 10: OTMA Destination Routing Descriptors

Before IMS Version 10, IMS systems that enabled OTMA and also produced ALTPCB outbound messages for external destinations required system programmers to code several assembler OTMA routing exits, including DFSYPRX0 & DFSYDRU0. IMS Version 10 introduced OTMA Destination Routing Descriptors that can eliminate the requirement to code the OTMA exits by externalizing the definitions and specifications that the exits provide. If the exits exist, they are called with the routing information that is provided by the descriptors that are already set.

Figure 4-10 describes an OTMA Destination Routing Descriptor.

- 'D' descriptor type in DFSYDTx member of IMS.PROCLIB

D destname keywords

Where:
Destname is destination name and can be masked by ending in an "*"

Keywords are:

```
TYPE={IMSCON|NONOTMA}  
TMEMBER=name  
TPIPE=name  
SMEM={NO|YES}  
ADAPTER=adapname  
CONVERTR=convname
```

- ◆ Up to 50 lines can be used in the specification of a descriptor
 - Columns 1 through 10 must be the same for each line of a continuation

Figure 4-10 Details of the D descriptor type

The 'D' descriptor type for the DFSYDTx member of IMS.PROCLIB has the following keywords:

- ▶ **TYPE**
Determines whether the output is destined for IMS Connect (IMSCON) or non-OTMA (NONOTMA). This is a required keyword.
- ▶ **TMEMBER**
A 1 - 16 character client name. This keyword is required for TYPE=IMSCON. It is ignored for TYPE=NONOTMA.
- ▶ **Tpipe**
A 1 - 8 character Tpipe name. It is optional for TYPE=IMSCON, where it defaults to destination name. It is ignored for NONOTMA.
- ▶ **SMEM**
Indicates whether this destination is a Super Member. It is an optional keyword for TYPE=IMSCON, where it defaults to SMEM=NO. It is ignored for TYPE=NONOTMA. If set to "YES", the name that defined in the **TMEMBER** keyword becomes the Super Member name and can be only four characters.
- ▶ **ADAPTER**
A 1 - 8 character name of the IMS Connect Adapter that is used for the message, for example, one example is an adapter for XML transformation. It is optional for TYPE=IMSCON and ignored for TYPE=NONOTMA.
- ▶ **CONVERTR**
A 1 - 8 character name of the Converter that is used by the adapter. It is required for TYPE=IMSCON if **ADAPTER** is specified. It is ignored for TYPE=NONOTMA.

Multiple OTMA descriptors can be defined in the same DFSYDTx member.

Value: Eliminates the requirement to code the OTMA exits by externalizing the definitions and specifications that the exits provide.

4.9.2 IMS Version 13: DFSYPRX0 and DFSYDRU0 exits override the OTMA destination descriptor

A new flag, input field, and return code were added into the user exit parameter list in the DFSYPRX0 (OTMA Pre-Routing Exit Routine) exit routine. The address at +88 points to a location that stores the routing information that is defined in the descriptor for WebSphere MQ and IMS Connect. If the destination name matches a non-OTMA destination in the descriptor or it does not match any entry in the OTMA destination descriptor, this new field is zero. For the destination as IMS Connect, see the TMAMICON_DESCRIPTOR DSECT mapping for the detailed routing information. For the destination as WebSphere MQ, see the TMAMMQS_DESCRIPTOR DSECT mapping for the detailed routing information.

Similarly, OTMA Destination Resolution Exit Routine (DFSYDRU0) added a flag, input field, and return code into the user exit parameter list. The address at +100 points to a location that stores the routing information that is defined in the descriptor for WebSphere MQ and IMS Connect. If the destination name matches a non-OTMA destination in the descriptor or it does not match any entry in the OTMA destination descriptor, this new field is zero.

Value: These two exits now have more routing information to access WebSphere MQ and IMS Connect.

4.9.3 IMS Version 13: WebSphere MQ descriptors

If you are using OTMA with WebSphere MQ, you now can route asynchronous output to WebSphere MQ without having to code OTMA routing exits DFSYPRX0 and DFSYDRU0.

You can use OTMA destination descriptors to perform the following actions:

- ▶ Send OTMA asynchronous callout messages to non-OTMA destinations, such as SNA printers and terminals.
- ▶ Initiate a synchronous program switch in IMS Version 13 between two IMS application programs.
- ▶ Send asynchronous and synchronous callout messages to IMS Connect.
- ▶ Send asynchronous callout messages to WebSphere MQ.

Figure 4-11 shows how an OTMA descriptor can be used with an asynchronous callout message to WebSphere MQ.

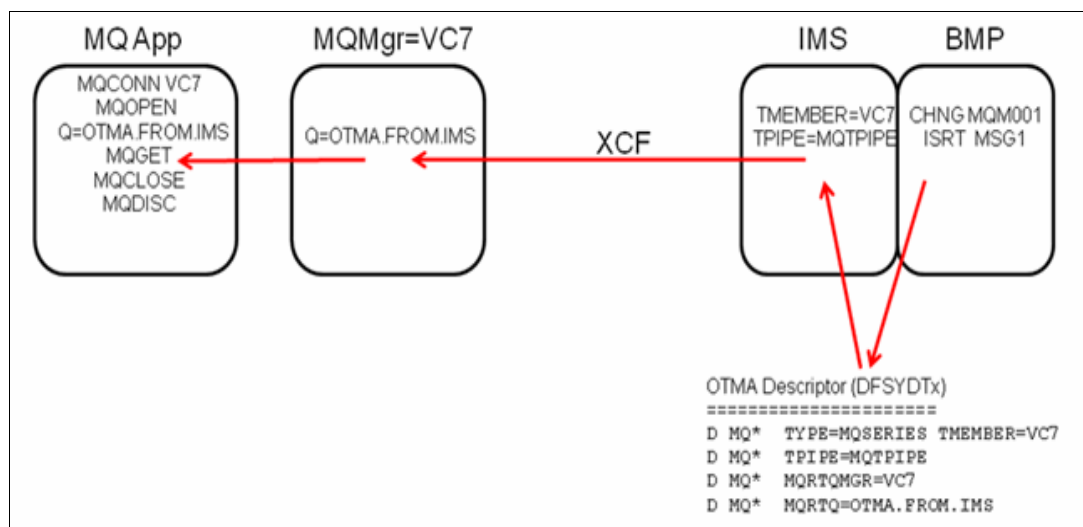


Figure 4-11 OTMA descriptor and asynchronous call messages to WebSphere MQ

In Figure 4-11, a BMP application program specifies MQM001 as the destination for an asynchronous callout message. An OTMA destination descriptor is defined with the name WebSphere MQ*. The asterisk in the descriptor name is a wildcard character.

When IMS receives the asynchronous callout message, IMS searches all OTMA destination descriptors to find a match for MQM001. In this case, IMS does not find a destination descriptor name that exactly matches MQM001, so the descriptor WebSphere MQ* is used. By using a wildcard character, one descriptor can be used for multiple destination.

Routing options for WebSphere MQ in the OTMA destination descriptor

To route asynchronous callout requests to WebSphere MQ, you must specify the following parameters in the OTMA destination descriptor:

- **TMBER**, to identify which OTMA client to send the message to.
- **MQRTQMGR**, to identify which WebSphere MQ reply queue manager processes the message.

Note: Because the **MQRTQMGR** parameter is optional in the DFSYDRU0 exit, it should be optional in the OTMA destination descriptor. With IMS Version 13 APAR PM94277, this parameter is an optional one.

- **MQRTQ**, to specify the WebSphere MQ queue for the message.

Figure 4-12 on page 161 lists the parameters that are available when coding a WebSphere MQ OTMA descriptor.

IMS V13 MQSeries OTMA Descriptors

- DFSYDTx PROCLIB member
 - IMS Cold start or /ERE COLDCOMM
- D MQ* TYPE=MQSERIES TMEMBER=VC7 TPIPE=MQTPIPE
- D MQ* USERID=USRT001 EXIT=NO SYNCTP=YES
- D MQ* MQRTQMGR=VC7 MQRTQ=OTMA.FROM.IMS
- D MQ* MQREPORT=NONE MQFORMAT=MQIMS MQPERST=YES
- D MQ* MQMSGID=MSG456789012345678901234
- D MQ* MQCORREL=COR456789012345678901234
- D MQ* MQAPPLID=APP45678901234567890123456789012
- D MQ* MQRTF=MQIMSVS MQCOPYMD=NO
- Required:
TMEMBER, MQRTQMGR, MQRTQ

Figure 4-12 IMS Version 13 WebSphere MQ OTMA Descriptor example

Figure 4-13 lists out the type-2 commands that are associated with OTMA descriptors.

IMS Type 2 OTMADESC commands

- QUERY, CREATE, DELETE, UPDATE
- Recovered across Warm and Emergency restarts
- QUERY SHOW(NONOTMA | IMSCON | MQSERIES | IMSTRAN)
 - Returns only fields applicable to that Type

Figure 4-13 IMS type-2 OTMADESC commands

Note: APAR PM90070 resolved a problem of asynchronous messages to WebSphere MQ using OTMA destination descriptors ending up on the Dead Letter Queue. PM90070 corrects this error by changing the MQMD value in OTMA destination descriptors to Version 1. The MQMD mapping in DFSYMSG has the Version 2 fields removed.

Value: Customers sending asynchronous output to WebSphere MQ can code simple descriptors rather than having to write assembler exits

4.10 OTMA scalability and performance

For multi-user web applications, reliable and efficient performance is a necessity. One key element that is associated with OTMA is ensuring that flood conditions do not degrade the OTMA capacity to effectively handle throughput. This section describes how OTMA handles these situations.

Another method to control the flow of transaction messages to the message queues is specifying a transaction expiration value. When transactions are not processing quickly enough, they can back up. If an expiration value is set, transactions that are not processed before that period expires are discarded, thus reducing processing and alleviating a message queue buildup. This topic is also examined in detail in this section.

4.10.1 Other general OTMA performance items

This section describes some other general OTMA performance items.

Dependent regions and CM0 OTMA transactions

If you use send-then-commit (CM0) mode transactions, they remain in the dependent region while the output is being sent (before the sync point occurs). Therefore, if many of your transactions are send-then-commit transactions, increase the number of dependent regions to improve throughput performance, or use as many commit-then-send (CM1) OTMA transactions as possible.

IMS message queue data sets and OTMA message prefixes

The OTMA message prefix, including the user data section, is stored on IMS message queue data sets, so if you are beginning to use OTMA, examine if the message queue data sets should be increased in size.

Virtual storage considerations

If an IMS OTMA environment has heavy OTMA traffic, a significant increase in LUMP and HIOP pool usage can occur. Because LUMP and HIOP pools are allocated from private storage, you might need to increase the size of the IMS control region address space. Also, certain OTMA control blocks are allocated from an extended common service area (ECSA), which is another limited resource.

OTMA security overhead

If possible, you can reduce security overhead in IMS by letting the clients perform all security checking. If you set the no-security-checking flag in the security-data section of the message prefix, you can avoid much of the RACF overhead and improve IMS performance.

Also, if RACF is used, the CHECK level includes a caching mechanism that can improve authorization checking performance.

4.11 OTMA scalability and performance enhancements

There have been major advances in the removal of inhibitors to OTMA scaling to manage high volume workloads. This section describes those advances.

4.11.1 IMS Version 10: Automatic removal of Tpipes

This clean-up enhancement determines whether an inactive Tpipe can be deleted and its storage released. OTMA removes eligible idle transaction pipes automatically during the IMS system checkpoint process if they are idle for two consecutive system checkpoints.

To be considered as being in an idle state, a Tpipe:

- ▶ May not process commit-then-send (CM0) messages in a shared queues environment.
- ▶ May not be STOPped.
- ▶ May not be the subject of a Tpipe trace.
- ▶ May not have any incomplete send-then-commit (CM1) messages.
- ▶ May not be using synchronous Tpipe from WebSphere MQ.
- ▶ May not have queued commit-then-send (CM0) output messages.

Value: The user has control over storage usage within the IMS control region where OTMA is.

4.11.2 IMS Version 10: Dependent region release after ACK or NAK timeout

To free dependent regions that are waiting for an ACK or NAK response, OTMA now performs timeout processing for send-then-commit (CM1) messages if the ACK or NAK response is not received from an OTMA client within the period that you specify. The OTMA client must be using an appropriate sync level value. After the ACK or NAK response times out, the dependent region is released to process other transactions.

Value: There is optimized usage of IMS dependent region resources.

4.11.3 IMS Version 12: OTMA SQ enhancement

IMS Version 12 supports the running of OTMA Commit Mode 1 transactions with a sync level of NONE or CONFIRM in a Shared Queues (SQ) environment using XCF communications, thus eliminating the need for Resource Recovery Services (RRS). The usage of synclevel=sync point still requires RRS.

Existing **A0S=** options in the DFSDCxxx member of IMS PROCLIB include the following ones:

- ▶ **A0S=N** specifies that shared message queue support for synchronous OTMA is not active. The default is inactive (N).
- ▶ **A0S=Y** specifies that shared message queue support for synchronous OTMA is active (Y) and uses RRS.
- ▶ **A0S=F** activates the function by force (F) even if another member in the IMSplex cannot activate the function.

The new options that support XCF include the following ones:

- ▶ **A0S=B** requests the usage of XCF to communicate between the FE and the BE systems for synchronous transactions with sync levels of NONE and CONFIRM.

The processing of requests that are associated with a synclevel of SYNCPT depends on the RRS specification. If **RRS=Y**, then transactions with the synclevel SYNCPT transaction can be processed at either a FE or BE IMS system using RRS. If **RRS=N**, then transactions with synclevel SYNCPT are processed only at the FE IMS.

- ▶ **A0S=S** requests the use of XCF to communicate between the FE and the BE systems for synchronous transactions with a sync level of NONE and CONFIRM along with RRS Multisystem Cascaded Transaction support for synchronous transaction with a sync level of SYNCPT.

Under this option, SYNCPT request processing is equivalent to **A0S=F**, meaning that if RRS is not active on one system, then the systems that have specified F (force) still queue incoming transactions globally (without any affinity.) Therefore, if a system without RRS tries to process one of these transactions, IMS abends the application with a U711 code.

- ▶ **A0S=X**, like **A0S=B** and **A0S=S**, requests the usage of XCF to communicate between the FE and the BE systems for synchronous transaction with sync levels of NONE and CONFIRM.

RRS is not used for synchronous transaction with a sync level of SYNCPT. In this case, the function of the **A0S=X** specification is equivalent to the **A0S=N** specification.

The B and X parameters are applicable only to the FE system. The BE system is not required to specify this parameter. The BE starts either the new XCF message processing or the existing RRS message processing, depending upon the message having the XCF indicator or the RRS indicator.

Value: This enhancement allows IMS to be the sync point manager with the usage of XCF, which simplifies sync point processing and eliminates unnecessary RRS overhead.

4.11.4 IMS Version 12: Reduced path length for OTMA transaction processing

Without this performance enhancement, OTMA experiences unnecessary processor impact because it always validates a Tpipe name against keyword names when each message is received. With this enhancement, OTMA performs tpipe validation only when a new Tpipe name is received. For the subsequent transactions using the same tpipe name, OTMA no longer performs tpipe name validation checking. Even for OTMA tmember clients like IMS Connect that use only one port tpipe for CM1 messages, validation checking is started only once rather than for every message from that port Tpipe.

Value: There is improved OTMA performance.

4.11.5 IMS Version 13: OTMA transaction expiration at GU time

OTMA transaction expiration at GU time is updated with two enhancements:

- ▶ By default, the symptom memory dump of U243 abend and its DFS554A message for an expired transaction are no longer generated. OTMA, however, can be configured to generate the symptom memory dump and DFS554A message by specifying TODUMP=YES in the DFSYDTx PROCLIB member or by setting the TMAMDUMP flag in the OTMA state data prefix of the input message.
- ▶ Also, by default, a DFS3688I message instead of DFS555I/DFS2224 is sent to the OTMA client at application GU transaction expiration. This support was introduced in Special Product Enhancement (SPE) APARs PM05984 (IMS Version 10) and PM05985 (IMS Version 11) and with the base level of IMS Version 12.

OTMA, however, now honors a request to return the original input message instead of the DFS3688I if the **TMAMINPT** flag is set in the OTMA prefix of the input message. WebSphere MQ V7.1.0 with APAR PM47795 supports this capability and sets the flag for requests where the **MQMD_REPORT** option returns the original input message.

Value: The value of this enhancement is to suppress symptom memory dumps and the DFS554 message for OTMA transaction expiration messages. This reduces processor cycles.

4.11.6 IMS Version 13: OTMA ALT-PCB output for shared queues

Shared queues environments can sometimes pose problems for ALTPCB output that is generated by a back-end IMS. By default, the back-end ALTPCB output is queued to the shared queues with an affinity to that system and must be retrieved from there. IMS Connect resolved this issue with its super member support.

Other OTMA member clients, however, like WebSphere MQ, that are connected to the front-end IMS cannot easily retrieve the message. IMS Connect also has this problem if super member support is not enabled.

This facility can override an SQ back-end affinity for ALTPCB messages. It allows ALTPCB output that is generated in a back end to be routed back to a front end for delivery.

There are two options:

- ▶ Specifying ALTPCBE= YES in the OTMA client descriptor in the DFSYDTx PROCLIB member of the front-end IMS system
- ▶ Setting the **TMAMALTB** flag (X'01') in the state data prefix of the input message

Figure 4-14 shows how you can override an SQ back-end affinity for ALTPCB messages.

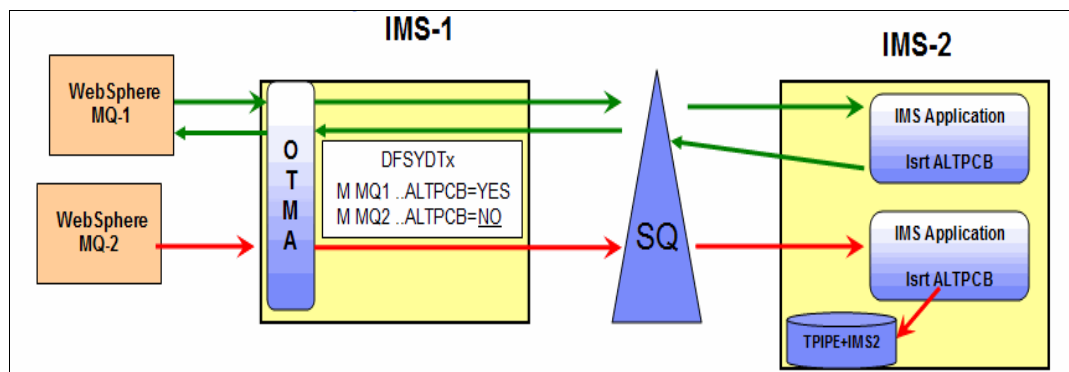


Figure 4-14 Override an SQ back-end affinity for ALTPCB messages

Value: This enhancement is valuable when super member support (for example, IMS Connect) is not used or not available (for example, WebSphere MQ).

4.11.7 Version 13: OTMA early termination notification

When OTMA is servicing high volume transaction systems, you want to minimize the messages that are sent to that OTMA instance during planned or unplanned terminations. The solution is that IMS OTMA disconnects from OTMA clients earlier in the termination process, thus providing an earlier notification to OTMA clients (such as IMS Connect and WebSphere MQ). The notifications are different based on whether the termination is planned or unplanned:

- ▶ When IMS has a planned shutdown, OTMA sends TMAMMNTR (x'3C' Resource Monitor) and TMAMCSPA (x'14' Suspend Tpipe) protocol messages to all OTMA clients.
- ▶ When IMS has an unplanned shutdown, OTMA leaves the XCF group earlier in the abend processing. The OTMA clients are notified through their XCF Group Exit.

Although this support is available in IMS Version 13 base code, it must be introduced into IMS Version 12 through SPE PM73869/UK9036 along with the pos-rec PM88737/UK94770.

Value: This enhancement provides earlier notification to OTMA clients that OTMA is no longer available.

4.11.8 IMS Version 13: MIPS reduction enhancement for YTIB hashing

Each YTIB block is associated with a commit mode 1 (CM1) input/output message or a commit mode 0 (CM0) input message. Today, all of the OTMA YTIB control blocks of a Tpipe are chained together and the Tpipe_TIB_HDR in DFSYPIP control block points to the first YTIB in the chain. When there are many YTIBs for a Tpipe, searching for a specific YTIB could take some processor cycles.

In IMS Version 13, a hash table with 23 entries at the end of a Tpipe structure for OTMA was created so that searching YTIB can be faster with the hashing logic. The existing Tpipe_TIB_HDR field is no longer pointing to a chain of YTIBs; it is now pointing to the beginning of the YTIB hash table with 23 hashing entries. This can have some vendor impact for the vendor products using Tpipe_TIB_HDR.

Value: Now the internal YTIB scan process is faster, which improves scan access by vendor tools.

4.11.9 OTMA client type notification

IMS OTMA clients (like IMS Connect, WebSphere MQ, and OTMA Call Interface (C/I)) each have unique processing considerations. IMS Connect is holdq-capable and supports the Resume Tpipe protocol. WebSphere MQ supports synchronous Tpipes for Commit Mode 0 (commit-then-send) messages. In many cases, these differences in client type impact how OTMA processed client requests and messages. The client initialization process (client-bid) is enhanced to allow clients to declare their “type” as OTMA.

IMS Connect in IMS Version 13 provides the OTMA client type at client-bid. OTMA C/I is identified as the internal DFSYICAL for Synchronous Program Switch. WebSphere MQ require changes to its client-bid to declare the client type.

Value: This enhancement allows OTMA to optimize certain initialization processes for greater efficiency.

4.12 Diagnostic tests that are related to OTMA

There are four main IMS facilities that you can use to diagnose problems with OTMA message traffic:

- ▶ OTMT table trace

To enable the OTMT trace table, run the following command:

```
/TRA SET ON TABLE OTMT OPTION LOG VOLUME HIGH
```

- ▶ To obtain the status of OTMA clients and servers, use **/DISPLAY** commands, such as **/DISPLAY OTMA**.

- ▶ OTMA tmember and tpipe traces

To enable the OTMA tmember trace, run the following command:

```
/TRA SET ON TMEMBER tmember name
```

If you know that the situation relates to a particular OTMA tmember, run the following command:

```
/DISPLAY TMEMBER tmember name Tpipe ALL
```


If you know that the situation relates to a particular OTMA Tpipe, run the following command:

```
/TRA SET ON TMBER tmember name Tpipe tpipe name
```

- For CM1 and CM0 problems that are associated with a specific PSB, enable DL/I tracing by running the following command:

```
/TRA SET ON PGM PSB name
```

After you enable the required traces, preserve the X'67D0' and X'6701' log records and the console output from the **/DISPLAY** commands.

4.13 OTMA diagnostic enhancements

OTMA diagnostic efforts rely on the base level IMS diagnostic facilities that are described in 4.12, “Diagnostic tests that are related to OTMA” on page 166, and tools such as the IMS Problem Investigator for z/OS, to trace event flows. There are some advances as new versions of IMS are introduced.

4.13.1 IMS Version 11: SQ BE transaction abend error message support

Before this capability existed, OTMA asynchronous support for Shared Queues environments documented a restriction relating to a back-end system abend situation. Transaction abends on a back-end system did not support sending the DFS555I message (application program abend) to the waiting remote client that is attached to a front-end IMS. The remote clients had to implement a timeout flow to ensure that the connection would eventually be broken.

Now this restriction is lifted to allow the DFS555I message to flow from the back-end IMS through the front end to the waiting remote client application.

Value: Provides consistency for IMS shared queues environments so that the result of the abend in either front-end IMS or back-end IMS is the same.



SOAP application technology

IBM IMS SOAP Gateway is a light-weight web services solution that enables IMS applications to interoperate in a service-oriented architecture (SOA) environment without needing a full-featured application server (for example, Java EE server). By using Simple Object Access Protocol (SOAP), IMS SOAP Gateway provides services that are independent of platform, environment, application language, or programming models.

This chapter provides a high-level understanding of IMS SOAP Gateway and how IMS SOAP Gateway fits into the overall IMS architecture. Also, this chapter introduces several new enhancements that are now available with IMS Enterprise Suite SOAP Gateway V2.2.

This chapter provides an SOAP Gateway overview and the new functionality and requirements for SOAP Gateway V2.2:

- ▶ Advanced installation support
- ▶ Monitoring, tracking, and logging
- ▶ SOAP Gateway message processing events
- ▶ COBOL top down and multiple hosts
- ▶ WS-Security for synchronous callout and migration

This chapter also provides an overview about what is in SOAP Gateway V3.1, the new feature in IMS Enterprise Suite Version 3.1:

- ▶ 64-bit support for z/OS
- ▶ Send-only with ACK support for synchronous callout
- ▶ SOAP Gateway management utility batch mode support

This chapter covers the following topics:

- ▶ IMS Enterprise Suite SOAP Gateway
- ▶ IMS SOAP Gateway overview
- ▶ IMS SOAP Gateway and your IMS applications
- ▶ What is new in IMS SOAP Gateway V2.2
- ▶ Asynchronous callout with SOAP Gateway
- ▶ Synchronous callout with IMS SOAP Gateway
- ▶ Multisegment support
- ▶ Security enhancements
- ▶ IMS SOAP Gateway features and compatibilities

5.1 IMS Enterprise Suite SOAP Gateway

IBM IMS Enterprise Suite SOAP Gateway, previously a separate product named IMS SOAP Gateway, is a web services solution that integrates IMS assets in an SOA environment, as shown in Figure 5-1.



Figure 5-1 SOAP Gateway

SOAP Gateway enables IMS applications to interoperate outside of the IMS environment through SOAP to provide and request services that are independent of platform, environment, application language, or programming model. SOAP Gateway helps you transform your IMS applications to either web service providers or consumers.

When IMS applications are enabled as web services providers, different types of client applications, such as Microsoft .NET, Java, and third-party applications, can submit SOAP requests into IMS to drive the business logic of the COBOL applications.

When IMS applications are enabled as web service consumers, they can call out to any external web services and receive the response in the same or a different transaction.

SOAP Gateway can be used on IBM z/OS, Linux on System z, and Windows systems.

5.2 IMS SOAP Gateway overview

IMS SOAP Gateway enables your IMS applications to perform as web services providers and consumers.

Different types of client applications, such as Microsoft .NET, Java, and third-party applications, can submit SOAP requests into IMS. With the IMS Connect XML adapter, you can enable your IMS application to become a web service without changing the back-end IMS application.

IMS SOAP Gateway also enables your IMS application as a web services consumer. Your IMS applications can make a callout request to access external web services providers and get responses back either synchronously or asynchronously.

IMS SOAP Gateway is compliant with the industry standards for web services, including SOAP/HTTP 1.1 and Web Services Description Language (WSDL) 1.1. This compliance enables your IMS assets to interoperate openly with various types of applications.

The IMS SOAP Gateway has several different components:

- ▶ **Connection bundle:** This specifies the connection and security properties between IMS SOAP Gateway, IMS Connect, and IMS. With the correlator file, this information is passed to IMS Connect from IMS SOAP Gateway. The connection bundle is generated by the IMS SOAP Gateway Management Utility. Here are the property options for the connection bundle:
 - Connection bundle name
 - IMS Connect host name
 - IMS Connect port
 - IMS Connect Datastore
 - User ID
 - Password
 - Groupname

Here are the settings for SSL support between IMS SOAP Gateway and IMS Connect:

- SSL Keystore name
- SSL Keystore password
- SSL Truststore name
- SSL Truststore
- SSL Encryption level

Here are the settings for HTTPS/SSL support between an external web server and IMS SOAP Gateway for the callout scenario:

- `<calloutSslKeystoreName></calloutSslKeystoreName>`
- `<calloutSslKeystorePasswd></calloutSslKeystorePasswd>`
- `<calloutSslTruststoreName></calloutSslTruststoreName>`
- `<calloutSslTruststorePasswd></calloutSslTruststorePasswd>`

For callout support, it is necessary to indicate the TPIPE names.

Example 5-1 shows a sample connection bundle.

Example 5-1 Connection bundle example

```
<conn>
<connBundleName></connBundleName>
<connHostName></connHostName>
<connPortNum></connPortNum>
<connDataStoreName></connDataStoreName>
<connUserID></connUserID>
<connPassword></connPassword>
<connGroupName></connGroupName>
<sslKeystoreName></sslKeystoreName>
<sslKeystorePasswd></sslKeystorePasswd>
<sslTruststoreName></sslTruststoreName>
<sslTruststorePasswd></sslTruststorePasswd>
<sslEncrypType></sslEncrypType>
<calloutTPipes></calloutTPipes>
<calloutSslKeystoreName></calloutSslKeystoreName>
```

```

<calloutSslKeystorePasswd></calloutSslKeystorePasswd>
<calloutSslTruststoreName></calloutSslTruststoreName>
<calloutSslTruststorePasswd></calloutSslTruststorePasswd>
<calloutBasicAuthName></calloutBasicAuthName>
<calloutBasicAuthPasswd></calloutBasicAuthPasswd>
</conn>

```

- **Correlator file:** An XML file that can be generated by Rational Developer for System z or the IMS SOAP Gateway Management Utility. It specifies transaction, runtime properties, and information that the IMS SOAP Gateway needs to match incoming requests to the appropriate back-end IMS application. After IMS Connect receives the user specifications from the correlator file, it uses the specification to know how long to wait, and so on. It also identifies the connection bundle. Here are the property options for the correlator:

- Connection bundle name
- Socket timeout
- Execution timeout
- LTERM name
- IMS transaction code
- Program Name
- XML Adapter type
- XML converter name

Example 5-2 shows an example correlator file.

Example 5-2 Correlator file example for IMS SOAP Gateway

```

<?xml version="1.0" encoding="UTF-8"?>
<COR:correlator mode="call_in" INSTALLATION="" UUID=""
  version="2.0" xmlns:COR="http://www.ibm.com/IMS/Correlator"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.ibm.com/IMS/Correlator correlator.xsd ">
  <wsdlFile name="IMSPHBK.wsdl"
targetNamespace="file://target.files1300386322968"/>
  <serviceTraceLevel>Error</serviceTraceLevel>
  <correlatorEntry operationName="IMSPHBKOperation" portName="IMSPHBKPort"
serviceName="IMSPHBKService">
    <adapterType>IBM XML Adapter</adapterType>
    <converterName>IMSPHBKD</converterName>
    <connectionBundleName>IMSPHBK</connectionBundleName>
    <socketTimeout>0</socketTimeout>
    <executionTimeout>0</executionTimeout>
    <ltermName></ltermName>
    <inboundTPIPName></inboundTPIPName>
    <inboundCCSID>1208</inboundCCSID>
    <hostCCSID>1140</hostCCSID>
    <outboundCCSID>1208</outboundCCSID>
    <trancode>IVTNO</trancode>
    <calloutConnBundleNames></calloutConnBundleNames>
    <calloutWSTimeout></calloutWSTimeout>
  <WSecurity></WSecurity>
    <calloutURI></calloutURI>
    <extendedProperty1></extendedProperty1>
    <extendedProperty2></extendedProperty2>
  </correlatorEntry>
</COR:correlator>

```

- **Callout connection bundle name:** The name of the callout connection bundle that contains the connection and security properties that are used to connect to IMS.
- **WSDL file:** An XML document that describes a web service. WSDL files are used by others (for example, the client that starts the service) to discover the service and to understand how to start the service. It specifies the location of the service and the operations that the service exposes. WSDL files are generated by Rational Developer for System z.

The WSDL file name, service name, and operation name pertain to both provider and callout scenarios. For the provider, they are the service details for the exposed IMS transaction. For the callout, they are the external web service details that IMS is trying to callout to.

- **Web services:** These are services, usually some combination of programming and data, which are made available for web users or other web-connected programs.
- **Server properties file:** This file is used to configure the IMS SOAP Gateway Server runtime environment. It is generated by the SOAP Gateway Management Utility.
- **Management Utility:** This utility is provided by IMS SOAP Gateway. It generates the connection bundle, the correlator file, and the server properties file. The Management Utility runs in the Windows DOS command prompt.

For those of you who are familiar with the previous version of SOAP Gateway, it is now called the Management Utility in Version 2.2. Figure 5-2 shows the new style on a Windows installation.

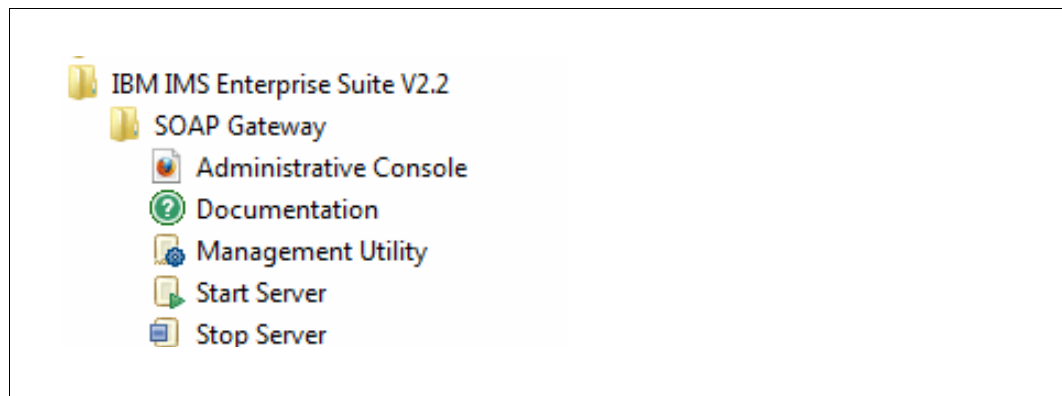


Figure 5-2 New Management Utility

- **IMS Connect XML Adapter:** The IMS Connect XML Adapter receives XML requests from IMS Connect. The XML Adapter then sends the XML message to an XML conversion function that can transform the XML request message into an application data format that the IMS COBOL application expects.
- **IMS Connect XML Converter:** The XML converters that are called by the XML Adapter are specific to each COBOL application. The converters can be generated by IBM Rational Developer for System z by using the COBOL copybook of the COBOL application. The IMS SOAP Gateway specifies to IMS Connect which XML converter to use for data transformation for the intended IMS COBOL application. IMS SOAP Gateway also supports PL/I.

We now provide examples of the flow of control that occurs with IMS SOAP Gateway solutions.

5.3 IMS SOAP Gateway and your IMS applications

IMS SOAP Gateway can enable your IMS applications as both a consumer and provider. Figure 5-3 shows the flow of control that is associated with an IMS application as a service provider.

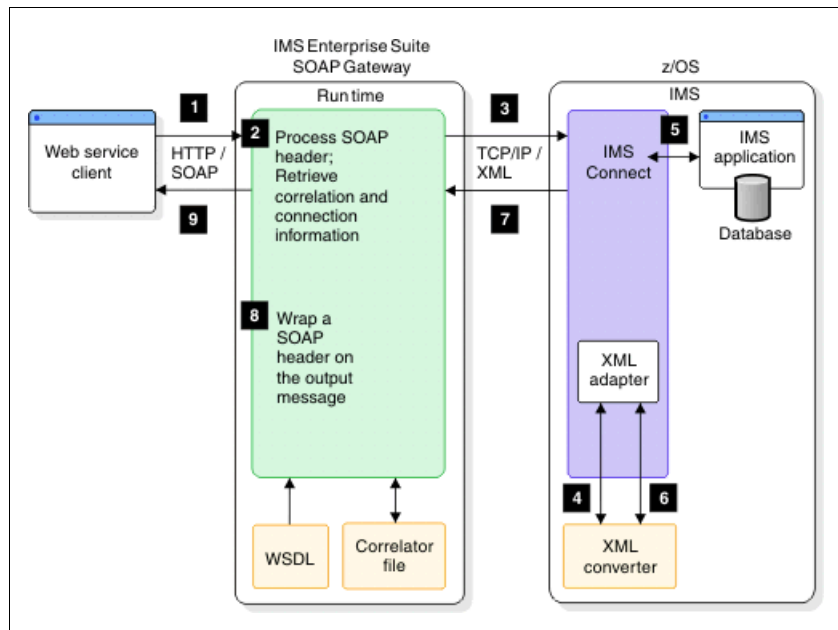


Figure 5-3 Flow of control with an IMS application as a web services provider

Here is the flow of control that is associated with an IMS application as a service provider:

1. The web services client application sends a SOAP message to IMS SOAP Gateway.
2. IMS SOAP Gateway processes the SOAP header (XML) and retrieves the appropriate correlation and connection information for the input request.
3. IMS SOAP Gateway sends the input XML data to IMS Connect using TCP/IP after adding the appropriate IMS Connect header.
4. IMS Connect calls the XML Adapter, which in turn calls the XML Converter to perform the XML to IMS application format transformation.
5. IMS Connect then sends the message to IMS for further processing.
6. IMS Connect calls the XML Adapter to transform the IMS application format response into XML.
7. IMS Connect sends the output XML message back to IMS SOAP Gateway using TCP/IP.
8. IMS SOAP Gateway wraps a SOAP header on the output message and sends it back to the client application.

IMS SOAP Gateway can also be used in environments where IMS applications are a web services consumer that requests services. Sections 5.5, “Asynchronous callout with SOAP Gateway” on page 199 and 5.6, “Synchronous callout with IMS SOAP Gateway” on page 207 detail these processing flows.

5.3.1 XML-to-bytes and bytes-to-XML

The XML Adapter supports translation between XML and IMS messages. The IMS Connect XML Adapter was created to give support to IMS Connect clients such as IMS SOAP Gateway. For inbound messages, IMS Connect starts the XML Adapter to translate the message for IMS, remove XML tags, and, if necessary, convert from Unicode to EBCDIC. For outbound flows, IMS Connect starts the XML Adapter to prepare an XML message and creates the XML tags. Also, if necessary, it converts the message from EBCDIC to the appropriate Unicode encoding schema.

Rational Developer for System z supports the generation of XML converters for COBOL or PL/I applications using COBOL copybooks or PL/I include files. The XML Adapter support in IMS Connect with the generated Rational Developer for System z XML converters, facilitates the conversion of XML transactional requests into byte stream application data structures, and vice versa. Each IMS application expects its messages to be in a certain data structure, therefore one XML Converter is needed for each IMS application.

If you choose to handle the data transformation in your application without using the IMS Connect XML Adapter, you do not need to start the XML Adapter. In this case, the incoming XML message is sent to IMS Connect and then to the IMS application. The same is true in reverse. The IMS application creates an XML output message that is sent to IMS Connect, then directly to IMS SOAP Gateway, and lastly to the web services client. The problem with not using the XML Adapter is that the IMS application must handle converting byte data into the XML data structure because the IMS transaction code cannot be processed in XML format.

5.4 What is new in IMS SOAP Gateway V2.2

Here are the new functions in IMS SOAP Gateway V2.2:

- ▶ Advanced installation support
- ▶ Monitoring, tracking, and logging
- ▶ Management Utility Command for TranAgent
- ▶ COBOL Top Down and multiple hosts
- ▶ WS-Security for Synchronous Callout and Migration

To use these functions, you need the following prerequisites and tools:

- ▶ Software requirements
 - Enterprise Suite SOAP Gateway V2.2
 - IMS V11 or IMS V12 with integrated IMS Connect SPE (PM69983 or PM76333)
- ▶ Tools
 - IBM Rational Developer for System z V8.5.1 or later
 - Installation Manager V1.5.3 or later (on Windows)
 - Installation Manager on z/OS FMID HG1N140

5.4.1 Advanced installation support

The post-SMP/E installation process is enhanced to provide installation and maintenance flexibility through the usage of IBM Installation Manager for z/OS and the ability to install SOAP Gateway components into different directories or mount points. IBM Installation Manager for z/OS simplifies maintenance by providing the ability to install and upgrade the server by pulling components from a centralized repository, and this whole process is delivered through the SMP/E process. The configurable server architecture allows parts of the server architecture to remain in read-only mode. More disk space can be added with increasing web services demands without impacting the server operations.

Advanced installation support by using the IBM Installation Manager for z/OS

The IBM Installation Manager is a tool that you can use to install and maintain your software packages. On z/OS, the IBM Installation Manager runs as a UNIX System Services application, and can be started from the UNIX System Services shell, shell scripts, or from MVS batch jobs.

The IBM Installation Manager allows you to simplify installation and maintenance and achieve faster value through a simplified product installation, update, and uninstallation with integrated prerequisite and interdependency checking. This installation occurs after the SMP/E installation.

Note: If you already have IBM Installation Manager for z/OS processed by SMP/E on the driving system, ensure that you have at least applied PTF UK79476 to upgrade the Installation Manager installation kit to Version 1.5.3.

If you already have Installation Manager for z/OS installed, you may verify whether you are at least at Version 1.5.3. APAR PM834465 updates the IBM Installation Manager installation kit to Version 1.6.2. After this APAR is applied, new Installation Managers are created at Version 1.6.2. To upgrade existing Installation Managers, rerun the command to create the Installation Manager (either **installc**, **userinstc**, or **groupinstc**) from the updated installation kit. In addition, APAR PM83465 updates the sample installation job instructions to provide more information about user ID and group requirements for IBM Installation Manager.

5.4.2 The new installation architecture in IMS Enterprise Suite V2.2

Before Version 2.2 of the IMS Enterprise Suite, SOAP Gateway is installed under one directory. Everything from the program's executable binary files, to server properties file, log files, and users' web service artifacts, is installed together under one mount point, as shown in Figure 5-4. Installation of multiple copies of the SOAP Gateway server typically requires the use of MVS DUMP, copytree, or PAX, and it is difficult to install from one managed copy.

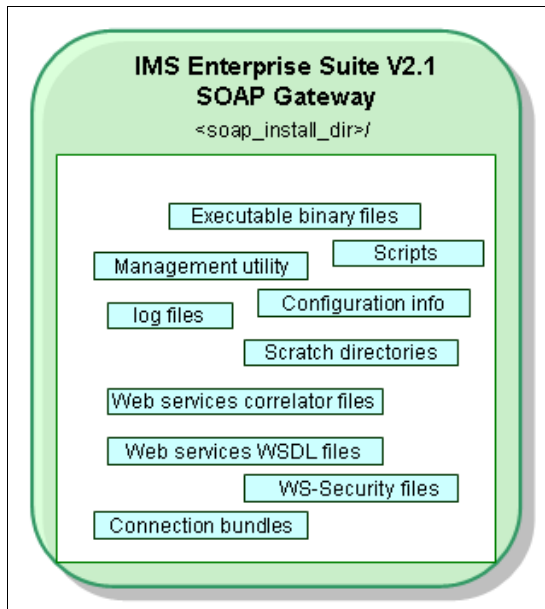


Figure 5-4 Before Enterprise Suite V2.2

Enterprise Suite before Version 2.2 has the following requirements:

- The directory where SOAP Gateway is installed must be writeable because users need to add web services, and the server log file is updated. Crucial server files therefore cannot be mounted in READ-only mode to prevent alterations.
- As more web services are deployed and the server continues to run, user files and log files can consume critical server space, and therefore slow down the server.

In Version 2.2, the SOAP Gateway installation is divided into three components:

- ▶ *imsserver* contains the server executable code, tools, and scripts. Files in this component should never be altered.
- ▶ *imsbase* contains the server configuration information, default log directory, and scratch directories.
- ▶ *imssoap* contains users' service artifacts, such as web services WSDL files, correlator files, connection bundles, and WS-Security configurations.

Figure 5-5 shows architecture of these components.

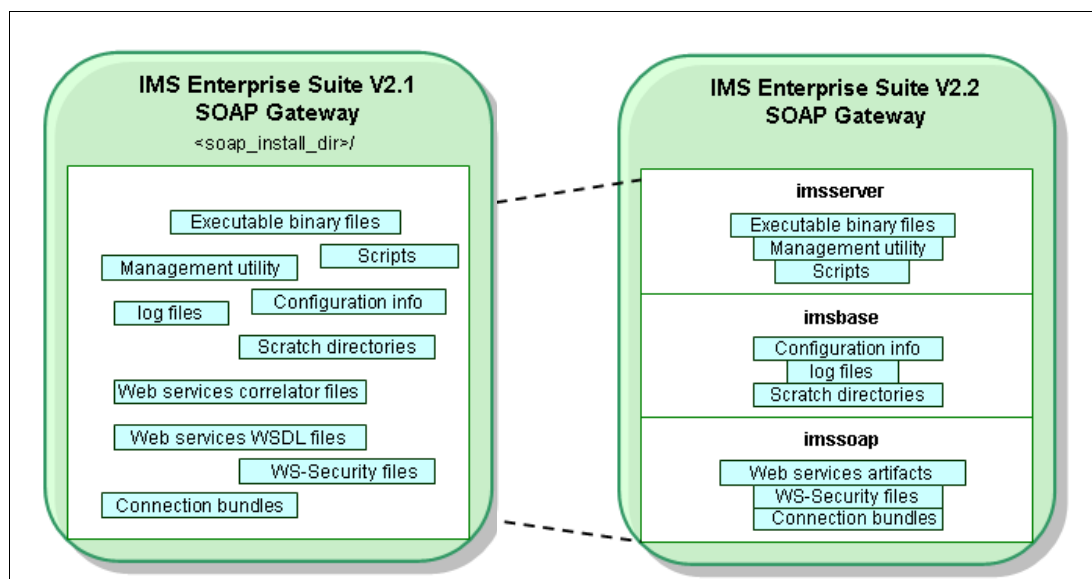


Figure 5-5 The new architecture

Each of the three components can be mounted separately, which means that the *imsserver* component can be mounted on a READ only data set for added security.

The *imsbase* and *imssoap* components must be mounted on a READ and WRITE data set. The data set volume depends on the number of services that you plan to deploy and whether logging is enabled. If more disk space is needed later, you can add more volumes and remount the *imsbase* and *imssoap* components without impacting the server component.

This three-part installation architecture enhances the server's security and its ability to grow with the increasing demands for web services.

This solves the first two issues with previous versions of SOAP Gateway, but the third issue, of installing multiple copies of SOAP Gateway from a managed copy, remains. The next section explains the usage of IBM Installation Manager for z/OS as a solution for this issue.

What is IBM Installation Manager for z/OS

IBM Installation Manager has been used on distributed platforms for many years for installing IBM products and applying updates. It installs from a repository that contains a `repository.config` file that stores the metadata on how to install and lay out the selected packages on the system.

More IBM z/OS products are using IBM Installation Manager for z/OS, including WebSphere Application Server for z/OS.

For more information about the installation process, see the video at the following website:

<http://www.youtube.com/watch?v=ui0ZKtyXJ8c>

5.4.3 Installing multiple copies of SOAP Gateway

Before you can install multiple copies of SOAP Gateway, you must run the SMPE installation of it.

For SOAP Gateway installation on z/OS, the SOAP Gateway repository file is a compressed file that is SMP/E-managed in one location. The Installation Manager for z/OS can connect to a repository through HTTP, FTP, or a shared disk. You then use the Installation Manager to install SOAP Gateway by pointing to the repository file.

You can install multiple copies of SOAP Gateway from the same copy of the SMP/E-managed repository. You can use the same instance of the Installation Manager to install SOAP Gateway on different LPARs if they are in the same Sysplex. If not, you must install the Installation Manager on that LPAR first.

From the tape or your Custom-built Product Delivery Option (CBPDO), you receive the Base Services and SOAP Gateway FMIDs. The IBM Installation Manager FMID is also included. Use the SMP/E installation process to obtain the sample jobs that are needed to install SOAP Gateway, the IBM Java SDK, and the SOAP Gateway repository. If you do not have IBM Installation Manager installed, use the SMP/E installation process to obtain the installation kit for the Installation Manager.

Figure 5-6 shows an overview of the SMP/E installation process.

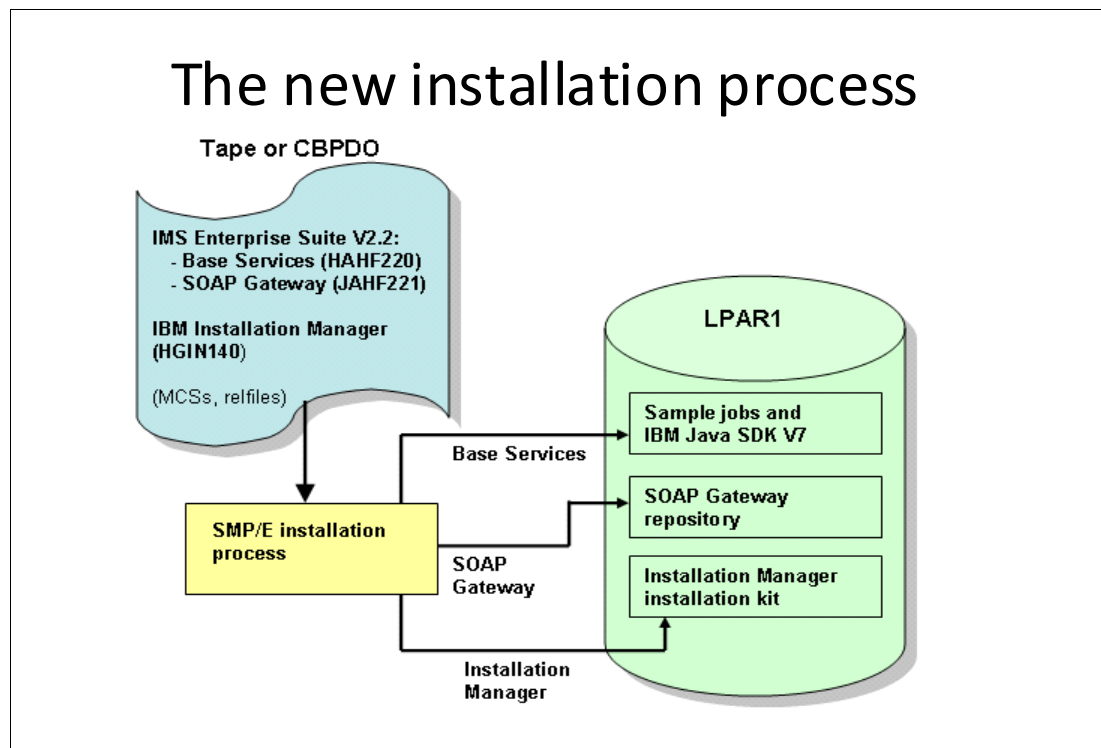


Figure 5-6 SMP/E installation process

After you RECEIVE, APPLY, and ACCEPT the FMIDs, the second step is to install the Installation Manager.

Before the installation, you must apply PTF UK79476 to upgrade to Version 1.5.3 of the Installation Manager. The version of the Installation Manager that is included in your order is an older version and it must be upgraded. Then, you can install the Installation Manager by editing and submitting the provided sample jobs.

After the Installation Manager is installed, you can use the Installation Manager Command Line **imc1** command to install SOAP Gateway by pointing to the SOAP Gateway repository.

Figure 5-7 shows an overview of the installation of multiple copies of SOAP Gateway.

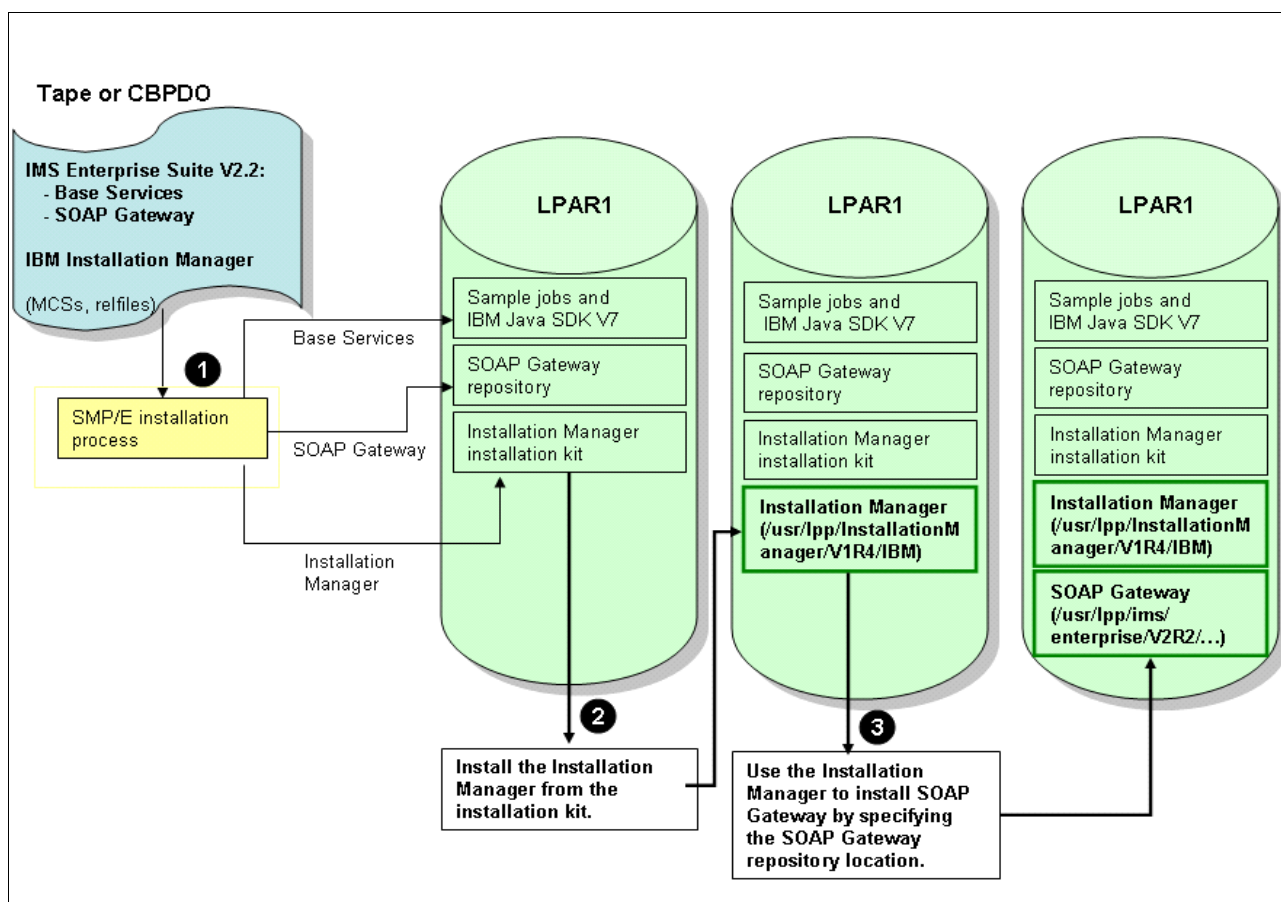


Figure 5-7 Multiple copies of SOAP Gateway

There are three installation components: imsserver, imsbases, and imsssoap. For each of these components, its installation directory or mount point must be specified.

Example 5-3 shows part of the AEWTSGCF sample job that is provided in the Base Services. You can edit this job and submit it to configure the installation directory. Instructions are provided in the sample job for the parameters that you must customize.

Example 5-3 AEWTSGCF sample

```
AEWTSGCF
/* 1) If you are installing SOAP Gateway on multiple mount */
/* points: */
/* a) Replace -ServerPathPrefix- with the path to install */
/* the imsserver component */
/* b) Replace -BasePathPrefix- with the path to install */
/* the imsbases component */
```

```

/*      c) Replace -SoapPathPrefix- with the path to install      */
/*      the imsssoap component                                    */
/*      These directories are created and mounted in sample job  */
/*      AEWTSGAL.                                                 */
ServerPathPrefix=-ServerPathPrefix-
BasePathPrefix=-BasePathPrefix-
SoapPathPrefix=-SoapPathPrefix-
IMPathPrefix=-IMPathPrefix-
RepoPath=ftp://driving.site.com/-RepoPathPrefix-
SharedResource=-SharedDirPath-

```

The Base Services include various sample jobs for SOAP Gateway installation for different installation scenarios, and they are documented in the SOAP Gateway documentation in the information center.

These are the configuration steps that you must follow before you can start SOAP Gateway:

► **AEWPARMF**

This is a sample job that copies the Java load module into the partitioned data set extended (PDSE) that is used by SOAP Gateway.

► **AEWIOGBP**

This is a sample job to specify the IBM Java SDK location and view the SOAP Gateway server properties by using the SOAP Gateway management utility commands from BPXBATCH.

► **AEWIOGCF**

This is a configuration member that configures SOAP Gateway runtime settings.

► **AEWIOGPR**

This is a JCL procedure for starting and stopping SOAP Gateway.

On the distributed platform, you use the Installation Manager, but the deployment utility has changed to Management Utility, as shown in Figure 5-2 on page 173. Also, the directory is the same described in Example 5-3 on page 180.

If you use UNIX and do not want to use JCL, you may use the OMVS and go to the imssserver mount point. From this mount point, you can perform the deployment as though you were using the Windows Management Utility.

Example 5-4 shows the usage of the `./iogmgmt` command to perform the deployment.

Example 5-4 Mount point imssserver/deploy on OMVS

```

deploy>./iogmgmt -view -sgp

```

A few notes about the **iogmgmt** command:

- To use the SOAP Gateway management utility, from the installation directory where the SOAP Gateway is installed (SOAP_Gateway_install_directory\imsserver\deploy), run **iogmgmt** (for Windows) or **./iogmgmt** (for z/OS and Linux on System z). Figure 5-8 shows the output of this command.

```

cicsts          ing          pdxt          wsrrPB3
IMSR2 § SC63:/Z1DRB1/usr/lpp>cd ims
IMSR2 § SC63:/Z1DRB1/usr/lpp/ims>ls
ims13q  imsb  imses  soap
IMSR2 § SC63:/Z1DRB1/usr/lpp/ims>cd soap
IMSR2 § SC63:/pp/imssoap/v2r2>ls
imsbase  imsserver  imssoap
IMSR2 § SC63:/pp/imssoap/v2r2>cd imsserver
IMSR2 § SC63:/pp/imssoap/v2r2/imsserver>ls
bin              icons              native.ais.properties
deploy           imsserverBackup_20130717_2205  server
docs             install             tools
IMSR2 § SC63:/pp/imssoap/v2r2/imsserver>dir deploy
dir: FSUM7351 not found
IMSR2 § SC63:/pp/imssoap/v2r2/imsserver>cd deploy
IMSR2 § SC63:/pp/imssoap/v2r2/imsserver/deploy>ls
commons-cli-1.2.jar          iogmgmt
log4j.deploy.properties
ims.esv2.2.1.iog.mgmt_20130314.jar iogmgmt.bat
IMSR2 § SC63:/pp/imssoap/v2r2/imsserver/deploy>./iogmgmt -view -sgp
IOG00018E: The command failed because the specified /pp/imssoap/v2r2/imsserver/java/bin
directory does not exist. The directory must
  already exist. Specify the full path to the Java directory, for example, C:ÖProgram
filesÖIBMÖIMS Enterprise Suite V2.2ÖSOAP Gatewa
yÖjava.
IMSR2 § SC63:/pp/imssoap/v2r2/imsserver/deploy>
===>

INPUT
ESC=Ä    1=Help    2=SubCmd    3=HlpRetrn  4=Top    5=Bottom    6=TS0    7=BackScr
8=Scroll  9=NextSess 10=Refresh
          11=FwdRetr 12=Retrieve

```

Figure 5-8 `deploy>./iogmgmt -view -sgp` output

- On Windows 7 systems, depending on the SOAP Gateway installation directory, you might need to open a command prompt as an administrator and change directories to the SOAP Gateway management utility directory (install_dir/imsserver/deploy) before you can run commands.
- An **iogmgmt** command consists of the **iogmgmt** statement followed by arguments to specify the command and associated options.
- Running a command with more than one instance of a parameter overrides all but the last instance of the parameter. For example, running **iogmgmt -corr -u -r MyCorr.xml -p MyService -i MyOperation -n NewBundleName1 -n NewBundleName2** sets the connection bundle name for the specified correlator entry to NewBundleName2.

- ▶ SOAP Gateway management utility commands are case-sensitive and must be entered in all lowercase or as shown in the command reference topic. For SOAP Gateway servers running on z/OS, the SOAP Gateway management utility must run on the same LPAR as the target server.

Table 5-1 shows the different paths after migration from a previous SOAP Gateway to Version 2.2.

Table 5-1 The different paths

ES V2.1 SOAP Gateway	ES V2.2 SOAP Gateway
server/webapps/imssoap/WEB-INF/classes/log4j.properties	imsbase/conf/log4j.properties
server/webapps/imssoap/imssoap.properties	imsbase/conf/imssoap.properties
N/A	imsbase/conf/native.env.properties
env.properties	N/A
server/conf/server.xml	imsbase/conf/master/server.xml
server/webapps/imssoap/xml/(correlators)	imssoap/xml/(correlators)
server/webapps/imssoap/wsd1/(wsdls)	Callout and business event: imssoap/wsd1/(wsdls)
server/webapps/imssoap/WEB-INF/conf/wsjaas.conf (a binary copy)	Provider: Not copied, but embedded in a service file
server/webapps/imssoap/WS-SECURITY/policy/bindings (a binary copy)	imssoap/WEB-INF/wsjaas.conf policy/bindings
server/webapps/imssoap/xml/connbundle (a binary copy)	imssoap/xml/connbundle.xml

5.4.4 Monitoring, tracking, and logging

This section describes the new IMS Transaction Tracking and monitoring support. Many clients need their enterprise system to be transparent, that is, when a request is sent from a distributed client to IMS, the path can be tracked.

So, when a response takes longer than usual to come back, what was the cause? To figure out the problem, you must discover where the bottle-neck is. The lack of end-to-end tracking capability for distributed requests to and from IMS must be addressed to discover the cause.

Another requirement in this area came from SOAP Gateway clients. As more clients use SOAP Gateway, system administrators must periodically monitor the “health” of SOAP Gateway. Simple metrics must be gathered to answer the following questions:

- ▶ When your system is running a load of thousands of transactions per second, is SOAP Gateway responding well at a certain point?
- ▶ What are the available services and their usage?
- ▶ Are the SOAP faults expanding to unacceptable levels?
- ▶ Is SOAP Gateway managing connections to IMS Connect well, or are connections growing beyond acceptable bounds?

The new version of SOAP Gateway provides an interface to gather such monitoring metrics.

Furthermore, there is a need to assist support users in performing problem determination on transactions. For example, the Internal Revenue Service asks for a specific record that was sent on a certain day and time, and this information must be available. To meet this demand, SOAP Gateway provides a transaction log that records the basic information for every request/response with minimal impact to performance.

Currently, the following restrictions apply to monitoring, tracking, and logging:

- ▶ Monitoring, tracking, and logging is supported only for provider scenarios. Callout support comes in releases after Version 2.2.
- ▶ Monitoring SOAP Gateway transactions is enabled only over an insecure JMX port. The JMX port that you listen to cannot be secured by SSL
- ▶ Tracking is supported only with IMS V12 and later.

SOAP Gateway V2.2 provides the following solutions:

- ▶ Publishing SOAP Gateway monitoring statistics (monitoring)
- ▶ Enabling transaction tracking from SOAP Gateway to IMS and back (tracking)
- ▶ Supporting problem determination through a transaction log (logging).
- ▶ Enabling ITCAM to process transaction event data.

SOAP Gateway Monitoring keeps statistics for requests by operation, noting operation details as successful, active, failed, total successful, and total failed. It does this by implementing a Java JMX MBean interface. Jconsole or client written applications can listen over the JMX port to get the data through the operations that are specified in the JMX interface.

Example 5-5 shows the command to enable SOAP Gateway monitoring from the Management Utility. The monitoring data is sent to port 7176.

Example 5-5 Monitoring that is sent to a specific port

```
iogmgmt -mbeans -on -port 7176
```

Transaction tracking

SOAP Gateway sends a tracking ID (up to 40 bytes) to IMS Connect on request and receives the tracking ID back on response.

The unique tracking ID is generated by SOAP Gateway unless a user-defined tracking ID is provided by the user, as shown in Example 5-6.

Example 5-6 Sample commands for tracking ID

```
iogmgmt -tracking -on|-off (-off by default) -id messageId|"generated"|"custom"  
messageID by default) -element e (element name needs to be specified only for  
custom) -nameSpace ns (ns is optional, can be specified only with custom)  
i.e.  
iogmgmt -tracking -on -id messageId
```

Configure SOAP Gateway to enable or disable tracking, and configure the tracking ID type.

In Figure 5-9, you see an IMS Transaction Tracking – Provider Scenario where the correlation is based on Tracking ID.

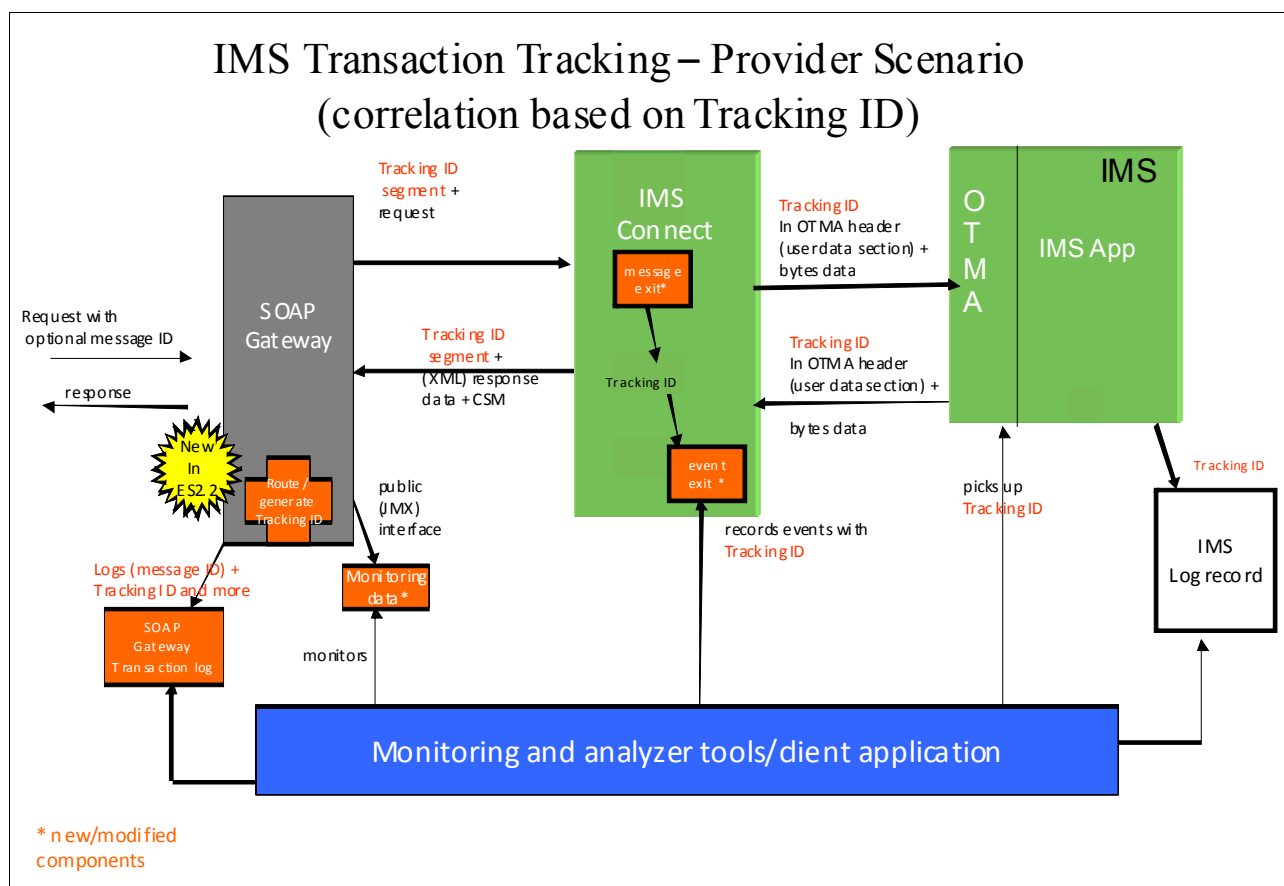


Figure 5-9 IMS Transaction tracking – Provider scenario

When it takes longer than usual for SOAP Gateway to send a response to a request message, there might be a performance bottleneck. The tracking ID feature can be used to pinpoint the problem and be used for performance analysis.

SOAP Gateway sends the tracking ID with the request to IMS Connect, which makes the ID visible through the event exit. The ID is also recorded in the IMS log record as it is placed in OTMA prefix user data section. On response, the data in OTMA user data section is echoed back out. IMS Connect returns the tracking ID on response back to SOAP Gateway. The inbound and outbound tracking ID is logged if the transaction log feature is enabled.

Tracking transactions from SOAP Gateway to IMS and back using logs

You see how the tracking propagates from the request to response, so now you can view some logs to see how the logs have propagated.

Figure 5-10 shows the SOAP Gateway Log.

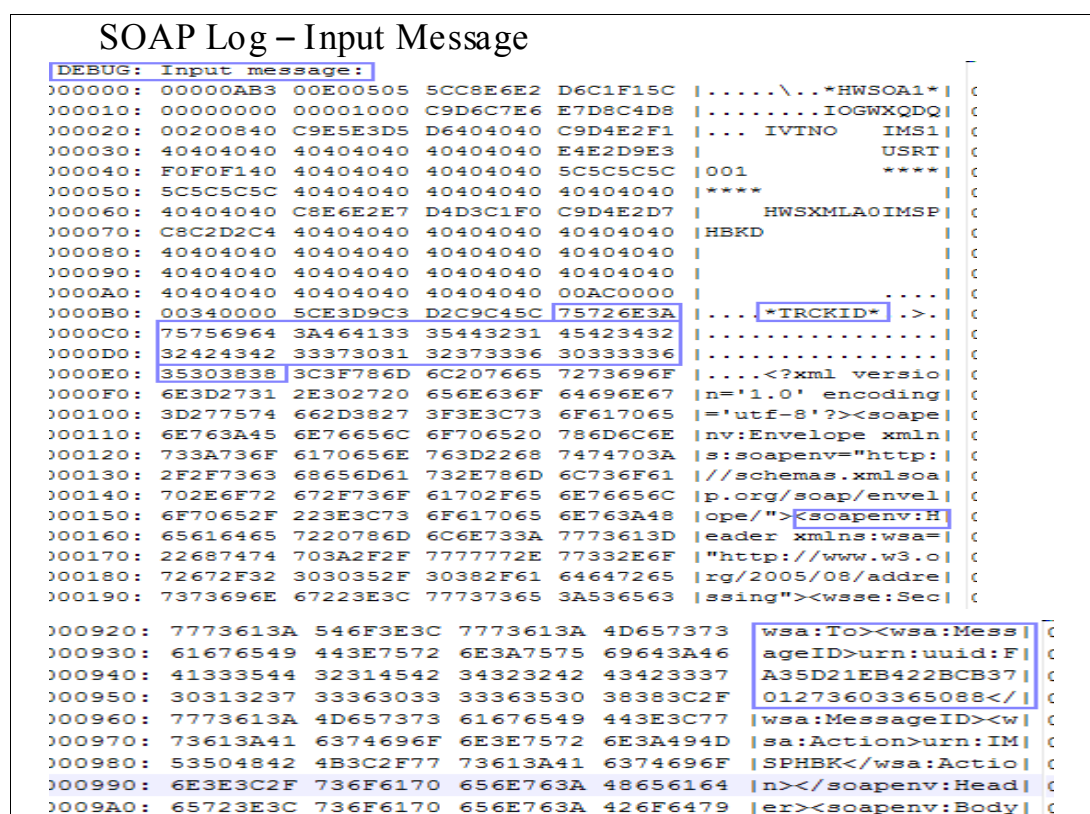


Figure 5-10 SOAP Gateway log of an input message

In the ICON BPE TRACE output, you find the following entries:

- ▶ ICONTR: The message that is received from a TCP/IP client.
- ▶ ICONIS: The message that is sent to IMS OTMA.
- ▶ ICONIR: The message that is received from IMS OTMA.
- ▶ ICONTS: The message that is sent to the TCP/IP client.

Figure 5-11 on page 187 shows the ICONTR entry of a BPE trace showing the trace and tracking ID.

Command input ==> █					SPB4.
	0	4	8	C	EBCDIC Data
0000	00000000	00000000	00000000	00000000	...3ICONTR
7614	00000000	00000000	00000000	5CC9D7C2r...o.....*IPB
F15C	00000000	00001000	C9D6C7E6	E7D8C4D8*HWSOA1*.....IOGWXQDQ
E2F1	40404040	40404040	40404040	E4E2D9E3	... IVTNO IMS1 USRT
5C5C	5C5C5C5C	40404040	40404040	40404040	001 *****
E2D7	C8C2D2C4	40404040	40404040	40404040	HWSXMLA0IMSPHBKD
4040	40404040	40404040	40404040	40404040*TRCKID*...>
0000	00340000	5CE3D9C3	D2C9C45C	75726E3A?/...>..._/_...%?/
3432	32424342	33373031	32373336	30333336	..?...?/...>...%?/...?/...
696F	6E3D2731	2E302720	656E636F	64696E67	./....._%>...../.....>.....?
7065	6E763A45	6E76656C	6F706520	786D6C6E?/...>..._/_...%?/
703A	2F2F7363	68656D61	732E786D	6C736F61	..?...?/...>...%?/...?/...
656C	6F70652F	223E3C73	6F617065	6E763A48	./....._%>...../.....>.....?
613D	22687474	703A2F2F	7777772E	77332E6F?/...>..._/_...%?/
7265	7373696E	67223E3C	77737365	3A536563?/...>..._/_...%?/
7365	3D226874	74703A2F	2F646F63	732E6F61?/...>..._/_...%?/
7373	2F323030	342F3031	2F6F6173	69732D32?/...>..._/_...%?/
6375	72697479	2D736563	6578742D	312E302E?/...>..._/_...%?/
7573	74556E64	65727374	616E643D	2231223E?/...>..._/_...%?/

Figure 5-11 ICONTR entry of a BPE trace

In the IMS input/output log record, you see an increase of x'34' bytes, which means the ICON User Data in the type01 type03 record was increased from the usual x'100' bytes to x'134' bytes, where the new/extra x'34' bytes is the *TRCKID* section.

Transaction logging

SOAP Gateway creates and produces versions of the transaction log.

The SOAP Gateway transaction log records information about every request that is made to a web services provider and the associated response messages from IMS.

With the commands shown in Example 5-7, you can configure SOAP Gateway to enable or disable transaction file logging, and configure the file properties.

Example 5-7 Configure and enable transaction file logging

```
iogmgmt -tranLog -on|-off
(-off by default) -filePath fp (fp is optional logging file path)
-fileNamePrefix fnp (fnp is optional logging file name prefix, default is
tranLog) -fileMaxAge fa (fa is optional, exclusive with fileMaxSize, default is
24) -fileMaxSize fs (fs is optional, exclusive with fileMaxAge, default is 1)

iogmgmt -tranLog -on -filePath C:\SOAPGateway\logs -fileNamePrefix
tranLog -fileMaxAge 24 -fileMaxSize 2
```

SOAP Gateway uses either a Vertical or Horizontal ID:

- ▶ Horizontal ID
 - It is available if you have tracking enabled.
 - It is propagated to IMS Connect, so you can use it to trace a specific request through SOAP Gateway, IMS Connect, IMS, and back.
- ▶ Vertical ID
 - It is always unique.
 - It is automatically generated and is not configurable.

5.4.5 SOAP Gateway message processing events

Each request message sent to the SOAP Gateway web service provider generates uniquely identified message events. An inbound request message records the following events:

- ▶ Event type 0: The server received a request from the client.
- ▶ Event type 10: The server validated the request message.
- ▶ Event type 12: SOAP Gateway sent the request message to IMS Connect.

After IMS processes the request message, it sends a response message back to the client application. The response message is associated with two additional event types:

- ▶ Event type 11: SOAP Gateway retrieved the response message from IMS Connect.
- ▶ Event type 17: The SOAP response message was sent to the client application.

The response can be a normal message response, a SOAP fault, an IMS Connect error response, an IMS error response, or an error from the target IMS application program.

Errors during message processing might cause a message to fail before all event types are recorded for the message. To determine why an error occurred, search the SOAP Gateway server log, IMS Connect log records, or IMS log records for the horizontal ID associated with the failed message.

In the transaction log shown in Figure 5-12 on page 189, we collected event types 0, 10, 12, 11, and 17.

Transaction Log

```
{
  "meta": {
    "version": "v1",
    "creation": 1355170855453
  },
  "event": {
    "instanceId": {
      "transactiondata": {
        "messageID": "urn:uuid:FA35D21EB422BCB3701273603365088",
        "hlink": "urn:uuid:FA35D21EB422BCB3701273603365088",
        "hlinkHex": "75726e3a757569643a46413335443231454234323242434233373031323733363033333635303838"
      },
      "verticalId": {
        "linkId": "c74eeb68129553875550b1d69446d9ea2e341d6ba5a2891e",
        "caller": "24",
        "timestamp": 1355170855.375000,
        "verticalContext": {
          "ComponentName": "IMS Enterprise Suite SOAP Gateway",
          "ApplicationName": "IBM-C76ECDEB3BC:8080",
          "ServerName": "IBM-C76ECDEB3BC",
          "type": 0
        }
      }
    },
    "instanceId": {
      "transactiondata": {
        "messageID": "urn:uuid:FA35D21EB422BCB3701273603365088",
        "hlink": "urn:uuid:FA35D21EB422BCB3701273603365088",
        "hlinkHex": "75726e3a757569643a46413335443231454234323242434233373031323733363033333635303838"
      },
      "verticalId": {
        "linkId": "c74eeb68129553875550b1d69446d9ea2e341d6ba5a2891e",
        "caller": "24",
        "timestamp": 1355170859.921000,
        "verticalContext": {
          "TransactionName": "file:\\\\target.files\\IMSPHBKService:IMSPHBKOperation",
          "type": 10
        }
      }
    },
    "instanceId": {
      "transactiondata": {
        "messageID": "urn:uuid:FA35D21EB422BCB3701273603365088",
        "User": "USRT001",
        "hlinkHex": "75726e3a757569643a46413335443231454234323242434233373031323733363033333635303838",
        "MessageLen": "2739"
      },
      "verticalId": {
        "linkId": "c74eeb68129553875550b1d69446d9ea2e341d6ba5a2891e",
        "caller": "24",
        "timestamp": 1355170860.359000,
        "type": 12,
        "horizontalContext": {
          "ComponentName": "IMS Connect",
          "TransactionName": "IVTNO",
          "ApplicationName": "ec03581.vmec.svl.ibm.com:9999:IMS1",
          "ServerName": "ec03581.vmec.svl.ibm.com",
          "horizontalId": {
            "linkId": "urn:uuid:FA35D21EB422BCB3701273603365088",
            "caller": "13"
          }
        }
      }
    },
    "instanceId": {
      "transactiondata": {
        "messageID": "urn:uuid:FA35D21EB422BCB3701273603365088",
        "hlinkHex": "75726e3a757569643a46413335443231454234323242434233373031323733363033333635303838",
        "MessageLen": "442"
      },
      "verticalId": {
        "linkId": "c74eeb68129553875550b1d69446d9ea2e341d6ba5a2891e",
        "caller": "24",
        "timestamp": 1355170861.281000,
        "type": 11,
        "horizontalContext": {
          "ComponentName": "IMS Connect",
          "ApplicationName": "ec03581.vmec.svl.ibm.com:9999:IMS1",
          "ServerName": "ec03581.vmec.svl.ibm.com",
          "horizontalId": {
            "linkId": "urn:uuid:FA35D21EB422BCB3701273603365088",
            "caller": "13"
          }
        }
      }
    },
    "instanceId": {
      "transactiondata": {
        "messageID": "urn:uuid:FA35D21EB422BCB3701273603365088",
        "hlink": "urn:uuid:FA35D21EB422BCB3701273603365088",
        "hlinkHex": "75726e3a757569643a46413335443231454234323242434233373031323733363033333635303838"
      },
      "verticalId": {
        "linkId": "c74eeb68129553875550b1d69446d9ea2e341d6ba5a2891e",
        "caller": "24",
        "timestamp": 1355170861.359000,
        "type": 17
      }
    }
  }
}
```

Figure 5-12 Transaction log

Each transaction record uses approximately 300 bytes of disk space, and each successful transaction generates five transaction records in the active log file

The SOAP Gateway transaction logger creates a record for every incoming request and outgoing response message that passes through the SOAP Gateway server

5.4.6 COBOL top down and multiple hosts

The previous versions of SOAP Gateway used the bottom up or meet in the middle approach. SOAP Gateway V2.2 uses the top down or WSDL first approach. The top down approach meets three needs:

- ▶ Quickly get a synchronous callout running with SOAP Gateway. This is useful for SOAP customers who want to have multi-operation support for synchronous callout.
- ▶ Removes the need for a COBOL copybook to generate artifacts for synchronous callout.
- ▶ Removes the requirement to map manually between XSD data types and COBOL data types.

The solution for meeting these needs is that the parts needed for synchronous callout can be generated from a single WSDL file. Also, multiple operations that are defined in a WSDL file are now supported within the callout correlators.

The top down approach eases the setup of synchronous callout for SOAP Gateway and reduces errors that might occur in a manually created COBOL copybook that is needed for a callout.

Figure 5-13 shows an overview of the flow.

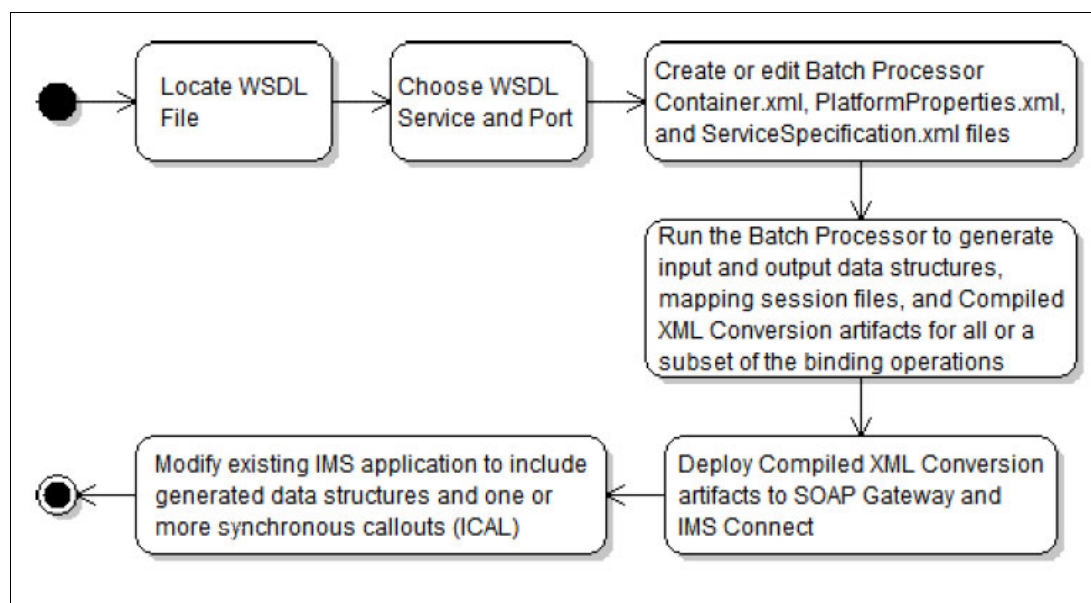


Figure 5-13 Design flow

Where:

1. Identify the WSDL service, port, and operations that your IMS application must start.
2. Create top-down Batch Processor options files (for example, `ServiceSpecification.xml`) that cite the WSDL service, port, and a subset of the operations.
3. Run the Batch Processor once to generate the following items for all (or a subset of) operations:
 - COBOL Copybook containing 01-level structures for each input or output message
 - COBOL XML converter for each operation
 - SOAP Gateway correlator containing information for each operation
 - Mapping session file for each input or output message

Here are the solution highlights:

- ▶ You create artifacts that are needed for synchronous callout from a single WSDL file. Multiple operations are supported for callout.
- ▶ Converters are generated for each operation that is defined in WSDL, and data types in XSD are associated according to COBOL data type.
- ▶ The correlator is populated with multiple operations if needed and multiple connection bundle support is built into IBM Rational Developer for System z V8.5.1.

Figure 5-14 on page 191 shows the operational specs that you must specify, such as `serviceName`, port, and other attributes in the `ServiceSpecification.xml` file

Define ServiceSpecification.xml

```
<ServiceImplementationSpec
  importDirectory=". /wsdl"
  importFile="HELLO.wsdl"
  wsdlServiceName="HELLOService"
  wsdlPortName="HELLOPort">
  <DriverSpec
    fileContainer="/converters"
    fileNamePrefix="HELLO"
    driverType="IMS_SOAP" />
  <CorrelatorSpec
    fileContainer="/correlator"
    fileName="HELLO.xml"
    adapterType="IBM XML Adapter"
    socketTimeout="50000"
    executionTimeout="50000"
    calloutWSTimeout="30000"
    calloutConnBundleNames="ch_host_ims1 ch_host_imsa" />
```

```
xsebatch -f C:\Hello\xsebatch_input\Container.xml -d C:\Hello\xsebatch_output -c -verbose
```

Figure 5-14 Operational considerations

The output is generated in the correlator, converter, and copybook by using the **xsebatch** command that is shown in Example 5-8.

Example 5-8 xsebatch that you need

```
xsebatch -s languageFile [-c | -w serviceName] | [-c -w serviceName] -f
containerFile [-d workspace] [-e WS_installdir] [-verbose] [-version]
[-overwrite=yes|no]
```

```
xsebatch -f C:\Hello\xsebatch_input\Container.xml -d C:\Hello\xsebatch_output -c
-verbose
```

Here are descriptions of the parameters that are used in Example 5-8:

► **-s languageFile**

This parameter is required unless the same information is specified in the importFile element. Here, languageFile is the language source file that contains the message definition. You can override this name by specifying values for the elements importDirectory and inputFile in the MessageSpec element in the ServiceSpecification.xml file.

► **-w serviceName**

This parameter causes the service definition files to be generated by using the specified name for the web service. You can override this option by using the **generateWSDL** option in the Container.xml file and in the ServiceSpecification.xml file. The value of this parameter can be overridden by the value attribute of the EISService element in the ServiceSpecification.xml file. The default value is set to esvc.

► **-f containerFile**

In Format (2), this parameter is required. It specifies the name of the Container.xml file that contains the generation options. Most of the elements in this file are optional, but a few are required and must be specified.

► **-d workspace**

This parameter specifies the fully qualified path of the workspace to be used for the import. If this path is not specified on the command line, then the default is taken from the environment variable `workspace`. If that environment variable is not set, then the default is set to the following settings:

- `$eclipse_root/workspace`
- `%eclipse_root%\workspace`

► **-e WS_installdir**

This parameter specifies the Eclipse subdirectory of the directory in which IBM Rational Developer for System z is installed. If the parameter is not specified, the default is taken from the environment variable `eclipse_root`. If that environment variable is not set, then the default is set to the default installation directory for IBM Rational Developer for System z. That directory is recorded during installation by the installation process and is set in the environment variable `WDZ71INSTDIR`.

► **-verbose**

This parameter causes the diagnostic messages to be printed to the console.

► **-version**

This parameter causes the version, release, and modification information to be printed to the console.

► **-novalidation**

This parameter turns off the validation of generation property files against an XML schema.

► **-overwrite=yes|no**

If set to `yes`, then this parameter causes newly generated files to overwrite existing files of the same name.

5.4.7 SOAP Gateway support for multiple hosts for callout

This function addresses SOAP Gateway customers who want to distribute workload between multiple hosts for callout web services and reduce the need for multiple SOAP Gateway services. It also removes the need to deploy duplicate IMS Connect XML converters over several hosts.

You can enable SOAP Gateway services to talk to multiple IMS Connect hosts or data stores. The distributed workload of callout over multiple hosts gives you the ability to share services and converters.

Figure 5-15 on page 193 shows a sample solution of a single SOAP Gateway with multiple IMS Connect hosts or data stores. The solution is composed of the following components:

- One SOAP Gateway
- One WSDL
- One correlator
- Two connbundles
- Two data stores or ICONs
- One descriptor
- One converter

SOAP Gateway multiple hosts

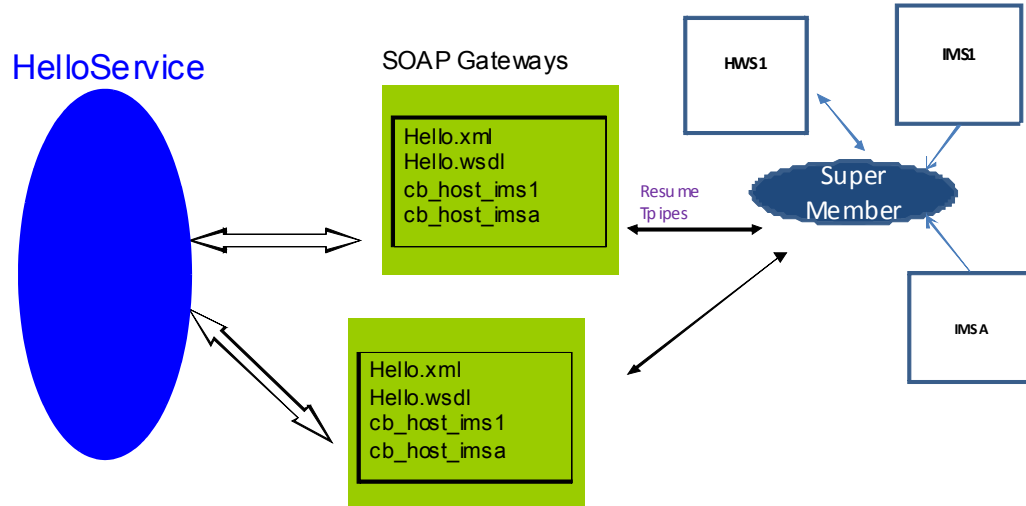


Figure 5-15 Multiple hosts

To enable SOAP Gateway support for multiple hosts for callout, complete the following steps:

1. Create a descriptor in DFSYTDx member for TMEMBER and XML Adapter, as shown in Example 5-9.

Example 5-9 DFSYTDx sample

```
D HELLO1 TYPE=IMSCON TMEMBER=SM01 SMEM=YES TPIPE=SGPSING9
D HELLO1 ADAPTER=HWSXMLAO CONVERTR=HELLOD
```

2. Compile and link a single converter.
3. Create two connection bundles by using the SOAP Gateway Management Utility, as shown in Example 5-10.

Example 5-10 Connection bundles

```
iogmgmt -conn -c -n cb_host_ims1 -d IMS1 -p 9999 -h myhost.com -i SGPSING9
iogmgmt -conn -c -n cb_host_ims1 -d IMSA -p 9999 -h myhost.com -I SGPSING9
```

4. Update the correlator to point to the two connection bundles, as shown in Example 5-11.

Example 5-11 Update the correlator

```
iogmgmt -u -r HELLO.xml -p HelloOperation -i HELLOService -d
cb_host_ims1,cb_host_imsa
```

5. Deploy the service by running the command that is shown in Example 5-12.

Example 5-12 Deploy the service

```
iogmgmt -deploy -w HELLO.wsdl -r HELLO.xml
```

The service should run, but if you run into a problem with running the service, there are several actions that you can take:

- ▶ Review the IMS job log to verify that the DFSYDTx data set is loaded upon IMS restart by running the following command:

```
READ SUCCESSFUL FOR DDNAME PROCLIB MEMBER = DFSYDTx
```

- ▶ Verify the MPP job log for ICAL and see whether the log has a return and reason codes of 0.

- ▶ Check the SOAP log for error messages that are related to the external web service:
 - If the external web service is not available, the SOAP log has the following message:

```
IOGS0077E:  
AN ERROR OCCURRED DURING THE INVOCATION OF THE EXTERNAL WEB SERVICE: THE  
SERVICE CANNOT BE FOUND FOR THE ENDPOINT REFERENCE
```

- If the external web service times out, the SOAP log has the following message:

```
IOGS0077E:  
AN ERROR OCCURRED DURING THE INVOCATION OF THE EXTERNAL WEB SERVICE: READ  
TIMED OUT
```

If you see these error messages, you should increase the Callout WS timeout value in the callout correlator. You also might need to increase the timeout value for ICAL in the IMS Application.

- ▶ A no tpipes specified case produces the following warning:
IOGX0019W: Skipping the callout connection bundle
- ▶ If the Connection Bundle is missing, you get the following warning:
IOGX0008W:The correlator file entries cannot processed

5.4.8 WS-Security for synchronous callout and migration

This section describes how to apply security to SOAP Gateway for synchronous callout. In the previous versions of SOAP Gateway, the authentication and authorization assertions were not propagated across web service transactions for callout.

In this example, we use the Originating Userid (PSTUSID) for IMS Synchronous callout application, which is passed to the external web service for further authentication and authorization. This application provides the message-level security for synchronous callout.

Figure 5-16 on page 195 shows the run time of a synchronous callout processing:

1. A callout application (in this sample, SYCALOUT.cbl) starts. The ICAL call in SYCALOUT.cbl specifies the name of the OTMA destination descriptor, which defines the name of the hold queue (or tpipes) in which the callout message is placed.
2. When the Hello callout web service is deployed, IMS Enterprise Suite SOAP Gateway polls the hold queue for callout messages.
3. IMS Connect converts the callout message from binary data to XML and sends it to SOAP Gateway.
4. SOAP Gateway sends the callout message to the external target server.
5. The target server sends a response.
6. SOAP Gateway sends the response back to the waiting IMS application.

Design Overview

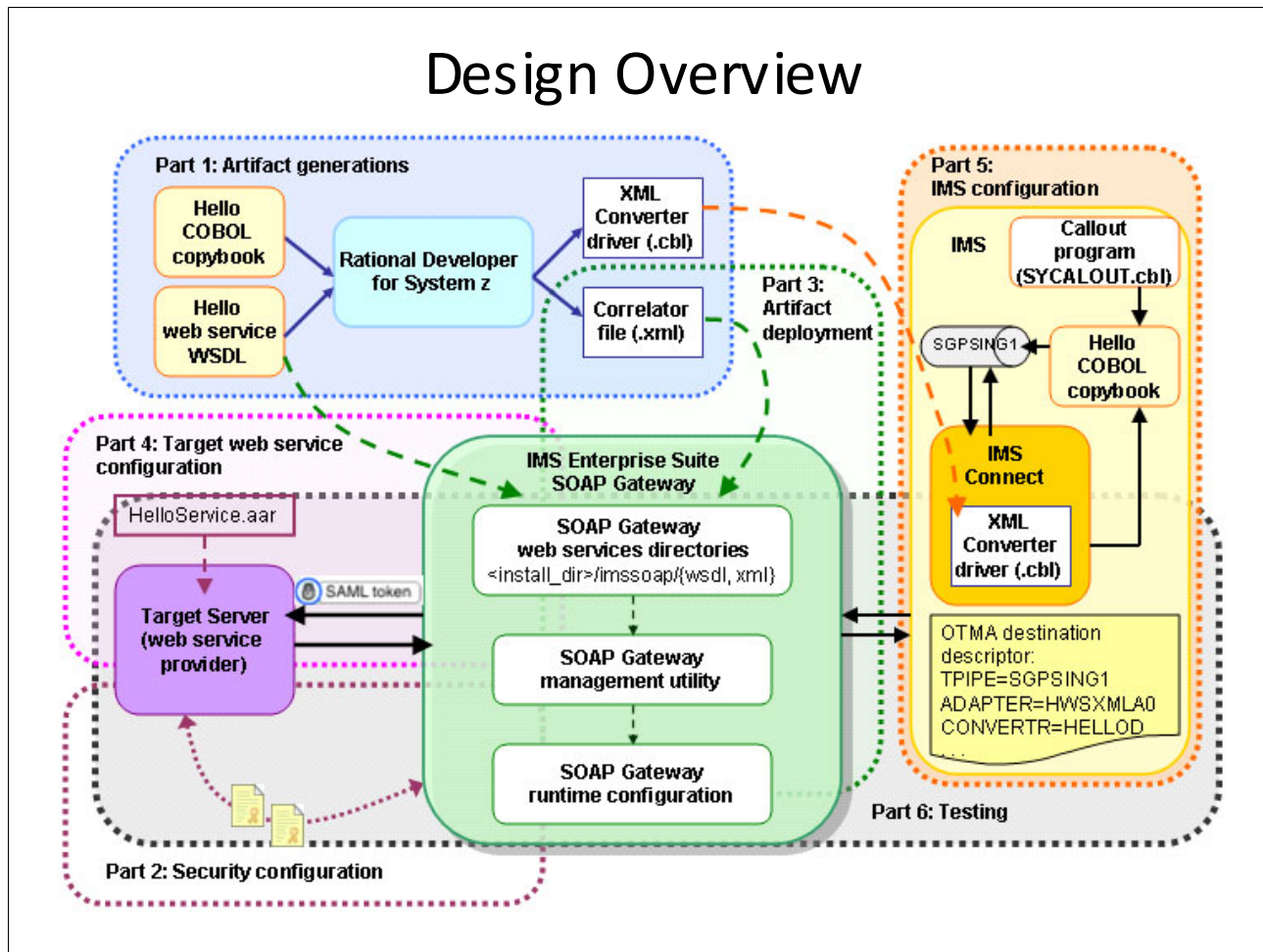


Figure 5-16 Run time of a synchronous callout processing

For the synchronous callout scenarios, in addition to transport-level security through basic authentication, server authentication, or mutual authentication, SOAP Gateway now supports message-level security with SAML 1.1 and SAML 2.0 sender-vouches unsigned tokens.

SAML is an XML-based standard that is developed by the Security Services Technical Committee (SSTC) of the Organization for the Advancement of Structured Information Standards (OASIS). This standard facilitates the following items:

- ▶ The exchange of user identity and security attributes information between communicating parties at the SOAP message level.
- ▶ The exchange of authentication and authorization assertions across web services transactions.

The WS-Security SAML confirmation method is supported by synchronous callout applications by extracting the user ID (the user that initiates the synchronous callout application) from the correlation token and passing it to the external web service.

SOAP Gateway also supports custom authentication modules for accessing the security header for validation before the SOAP request messages are sent out to the external web services server.

This support has the following considerations:

- ▶ Supports only WS-Security for Synchronous callout.
- ▶ Supports only SAML1.1 or SAML2.0 unsigned assertion.
- ▶ Uses different TPIPEs for security and non-security connections.
- ▶ If you use Rational Developer for System z V8.0.3.2 generated correlator files, then you must run the migration tool first to update the correlator files to the latest schema.
- ▶ Keystores/truststores for both the provider and callout cannot coexist in the same SOAP Gateway instance.
- ▶ All operations in a service must be the same security type.

To configure client authentication between SOAP Gateway (client) and external web services server, you must complete the following steps:

1. Use the Java keytool command to create a keystore or truststore for both the client and server, exchange the certificate, and import the other side certificate to truststore, for example, importing the client-side certificate to the server-side truststore.
2. Use management utility tool, (Example 5-13) to update the client-side (SOAP Gateway) keystore, truststore, and passwords information to the connection bundle file.

Example 5-13 Update client side

```
iogmgmt -conn -c -n MyCalloutConnBundle -h ICONHOST -p 9998 -d IMSSTOR1 -i  
tpipe1 -l /path/to/MyCalloutClientKeystore.ks -y MyCalloutClientKeystorePwd -v  
/path/to/MyCalloutClientTruststore.ks -q MyCalloutClientTruststorePwd
```

3. Deploy a callout web service with either the “SAML11Token” (Example 5-14) or “SAML20Token” token type.

Example 5-14 Deploy SAML11 token

```
iogmgmt -deploy -w hello.wsd1 -r hello.xml -t SAML11Token
```

The callout message is encrypted and decrypted during the transportation socket security; there is some performance impact.

Information that is needed to diagnose a problem

To collect information that is needed to diagnose a problem, turn on debug mode in the SOAP log by changing to level 5. You can change to level 5 by using the property command that is shown in Example 5-15.

Example 5-15 Debug mode

```
iogmgmt -prop -u -l 5
```

Here are some error scenarios and possible causes.

- ▶ In scenario 1, shown in Figure 5-17 on page 197, you receive the following error message:
CWSS6521E: The Login failed because of an exception:
javax.security.auth.login.LoginException: Unable to find valid certification
path to requested target

```
org.apache.axis2.AxisFault: CWWSS6521E: The Login failed because of an exception:
    javax.security.auth.login.LoginException:
        unable to find valid certification path to requested target
    at org.apache.axis2.util.Utils.getInboundFaultFromMessageContext(Utils.java:517)
    at org.apache.axis2.description.OutInAxisOperationClient.handleResponse(OutInAxisOperation.java:370)
    at org.apache.axis2.description.OutInAxisOperationClient.send(OutInAxisOperation.java:416)
    at org.apache.axis2.description.OutInAxisOperationClient.executeImpl(OutInAxisOperation.java:228)
    at org.apache.axis2.client.OperationClient.execute(OperationClient.java:163)
    at target.files.PL1TYPEServiceStub.emptyInRandomOutOperation(PL1TYPEServiceStub.java:816)
    at PL1TYPE_ADB_Security.main(PL1TYPE_ADB_Security.java:39)
CWWSS6521E: The Login failed because of an exception: javax.security.auth.login.LoginException
    unable to find valid certification path to requested target
```

Figure 5-17 Error scenario #1

Possible causes might be:

- You did not import the client-side certificate to the server-side truststore correctly.
- The truststore does not contain the root certificate.

Use the keytool command to check the certification command, as shown in Example 5-16.

Example 5-16 Check certificate

```
keytool -printcert -file absolute path to cert
```

- Scenario 2 shows a token problem, as shown in Figure 5-18.

```
<?xml version='1.0' encoding='utf-8'?><soapenv:Envelope
  xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"><soapenv:Header
    xmlns:wsa="http://www.w3.org/2005/08/addressing"><wsa:Action>http://www.w3.org/2005/08/
    /addressing/soap/fault</wsa:Action><wsa:RelatesTo>urn:uuid:DCA3E8B163DADD40EB13
    983896543</wsa:RelatesTo></soapenv:Header><soapenv:Body><soapenv:Fault><faultcode>
    >soapenv:Server</faultcode><faultstring>security.wsse:security.WSSContextImpl.s02:
    org.apache.axis2.AxisFault: CWWSS5502E: The target element: saml2:Assertion was not
    expected.</faultstring><detail /></soapenv:Fault></soapenv:Body></soapenv:Envelope>
```

Figure 5-18 Error scenario #2

Usually, the policy set is configured to require the token type; the received message does not have the token in it or it might be that you have SAML20Token configured, but SAML20Token is not being sent, so you received SAML11Token instead.

5.4.9 IMS Enterprise Suite V3.1 features

In IMS Enterprise Suite V3.1, SOAP Gateway adds the following support:

- 64-bit support for z/OS
- Send-only with ACK support for synchronous callout
- SOAP Gateway management utility batch mode support
- Enhanced security cipher suite support

64-bit support for z/OS

SOAP Gateway now runs on 64-bit on the z/OS platform, allowing organizations to take advantage of their 64-bit operating environment for extended memory usage.

Send-only with ACK support for synchronous callout

Send-only with acknowledgement protocol support for synchronous callout allows SOAP Gateway to receive a final confirmation that the response message was delivered to the original IMS application that issued the callout request. This confirmation provides SOAP Gateway users additional information about whether a callout response message was sent to IMS and whether IMS received the message.

Send-only with acknowledgement protocol for web services consumer applications

The send-only with acknowledgement protocol allows your application to receive a final confirmation that the response message was delivered to the original IMS application that issued the callout request. However, only synchronous callout applications can use the send-only with acknowledgement protocol.

By default, SOAP Gateway uses the send-only protocol without acknowledgement to send callout response messages to IMS. The send-only with acknowledgement protocol provides an alternative approach that offers an extra level of confirmation that the response message was received. After the callout response message is sent to IMS, IMS sends either an ACK (positive acknowledgement) or NACK (negative acknowledgement) response to SOAP Gateway. The ACK or NACK is not sent to the external callout target application, but SOAP Gateway logs the response.

For NACK responses, SOAP Gateway logs an IOGS0082E message if the trace level is set to error or higher. For ACK responses, SOAP Gateway logs an IOGS0081I message if the trace level is set to Informational or higher. If no ACK or NACK is received, SOAP Gateway logs an IOGS0083E message.

The main advantage of the send-only with ACK protocol is that you can easily identify if a callout response message was sent to IMS, whether IMS received the message, and how long it took the message to reach IMS. The main disadvantage is that the ACK or NACK reply adds an extra network communication event for each callout response message.

You can enable the send-only with acknowledgement protocol for a web service consumer application with the SOAP Gateway management utility when you create or update a correlator. Set the value for the `-k` property to `true`. The send-only with acknowledgement protocol is configured at the operation level and can be enabled or disabled without restarting the SOAP Gateway server.

SOAP Gateway management utility batch mode support

Administrators can now use the batch mode of the management utility to facilitate web services deployment and server management for better performance. Instead of issuing one command at a time, each with its own JVM instance, a file with a list of commands can be passed to the SOAP Gateway management utility batch command for execution in one JVM instance.

Enhanced security cipher suite support

SOAP Gateway is enhanced to use the FIPS 140-2 approved cryptographic providers IBMJCEFIPS (certificate 376) or IBMJSSEFIPS (certificate 409) for cryptography. The certificates are listed on the NIST website found at:

<http://csrc.nist.gov/cryptval/140-1/1401val2004.htm>

SOAP Gateway also adds support for Transport Layer Security (TLS) V1.2 and for cipher suites with key length of 2048 and key strength of 112 bit, as required by NIST SP800-131A.

5.5 Asynchronous callout with SOAP Gateway

There is an increasing need for core IMS business functions to access new business logic and data running outside the IMS environment. There is interest in transforming IMS applications into business-to-business (B2B) clients, not just components within a B2B server.

IMS Version 10 adds application asynchronous callout support to a web service through IMS SOAP Gateway. IMS callout refers to the ability of an IMS application to act as a client to start an application that is external to the IMS environment (for example, an application running in the distributed platform).

To support IMS asynchronous callout to web services, IMS SOAP Gateway implements the IMS Resume TPIPE protocol internally and manages a resume TPIPE loop inside the IMS SOAP Gateway server to continually retrieve asynchronous messages that were inserted from an alternative TPPCB by the IMS application. After the callout request message is received, IMS SOAP Gateway processes the request, finds the web service identifier in the callout request message, and matches it with the deployed web services to locate and start the web service. The started web service either performs the task and ends or sends the callout response message back to IMS to start a new IMS transaction, as shown in Figure 5-19.

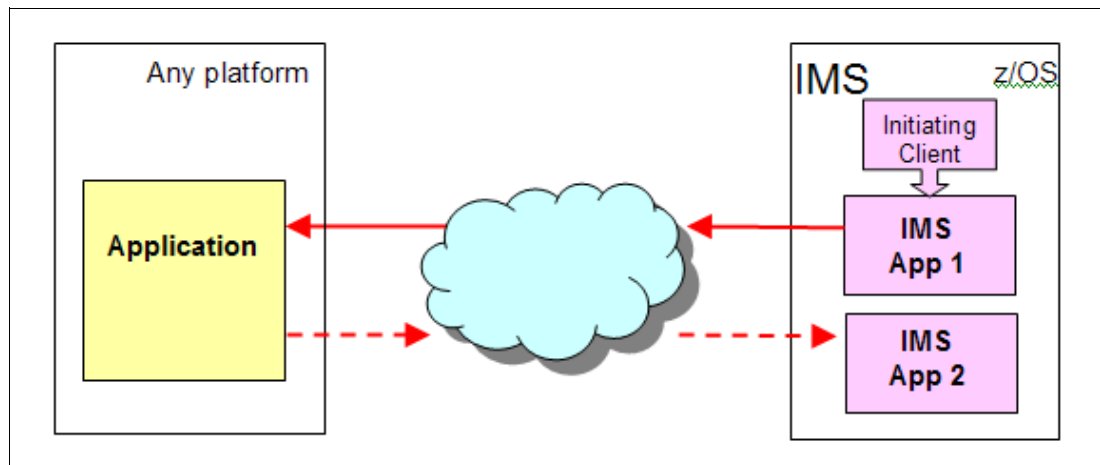


Figure 5-19 Asynchronous Callout flow

The advantage of using asynchronous callout is that it does not hold up the region, but it does require the IMS application to handle the output from the external application (if any) in a different IMS application instance.

5.5.1 Starting the web service operation

There are two types of web service operations for asynchronous callout: the one-way invocation and the request-response invocation. The one-way invocation corresponds to the one-way port type and the request-response invocation corresponds to the request-response port type of the web services operation. In the one-way invocation, no response is expected from the web service. In the request-response invocation, a response message is synchronously returned to the caller.

If the operation is one way, it sends out the request message by taking the XML message as the payload data but does not wait for the response to come back. If the operation is request-response, IMS SOAP Gateway waits for the response data to synchronously come back from the same SOAP/HTTP connection.

After the request message is sent successfully, IMS SOAP Gateway sends an ACK (positive acknowledgement) internally to IMS Connect to indicate that the callout request message is received and processed such that it can be removed from the TPIPE. In the case of the request-response request, the ACK is sent back to IMS Connect after the request is sent successfully to the web service but before IMS SOAP Gateway starts waiting for the response message from the web service.

If the send fails or a processing error occurred before starting the web service, a NAK (negative acknowledgement) with reroute command is sent to IMS Connect such that the callout request message is removed from the callout TPIPE and be rerouted to an error queue.

5.5.2 Returning the callout response message to IMS

In a request-response invocation, a callout response message is returned by the web service to the IMS SOAP Gateway. For asynchronous callout, this response message is sent back to the IMS system by starting a new IMS transaction, which we call Appl2 in our example.

When IMS SOAP Gateway receives the response message, it first extracts the response payload data from the SOAP message. Then, an IMS Connect message is built to send the response message back to the IMS Appl2. The IMS Connect message has the format, LLLLIRM<Response data>. IRLM is the transaction code for the IMS Appl2. It also contains the XML Adapter name and the converter name if the user chooses to use the XML adapter function to transform the callout response message. The transaction code, XML adapter, and converter name are all obtained from the correlator file that is specified by the user during the deployment step. If the user chooses not to use the XML adapter, IMS SOAP Gateway adds LL ZZ and transcode in front of the response data.

After the response message is built, it is sent to IMS Connect using Commit Mode 0 with Send Only protocol.

When IMS Connect receives the XML response message from the IMS SOAP Gateway, the XML conversion function converts the XML response data into the IMS application data format if the user chooses the XML adapter function. Upon receiving the converted application data bytes from the converter, the XML Adapter computes the new LL value. It also sets the transaction code value (obtained from the IRLM that IMS SOAP Gateway sends) within the message. The resulting response data message is then sent to IMS Appl2.

Note: If the callout response must go back to the initial client, a synchronous callout solution is recommended. However, a pseudo-synchronous solution can be achieved with the appropriate application design, for example, the initial IMS application can wait in the dependent region and continue to check for the output data (updated by a second transaction) through a database.

5.5.3 Web services callout scenarios for Asynchronous Callout

This section describes three common scenarios. All of these scenarios use the IMS Connect XML Adapter function for XML transformation and the OTMA descriptor function.

Scenario 1: Callout to one-way web service with no response to IMS

In this scenario, which is shown in Figure 5-20, the IMS application calls out to a one-way operation on the web service. Neither a response or error is expected from the web service. An example of this type of processing is when an IMS application needs to start a web service to perform a task, such as a callout to a printer.

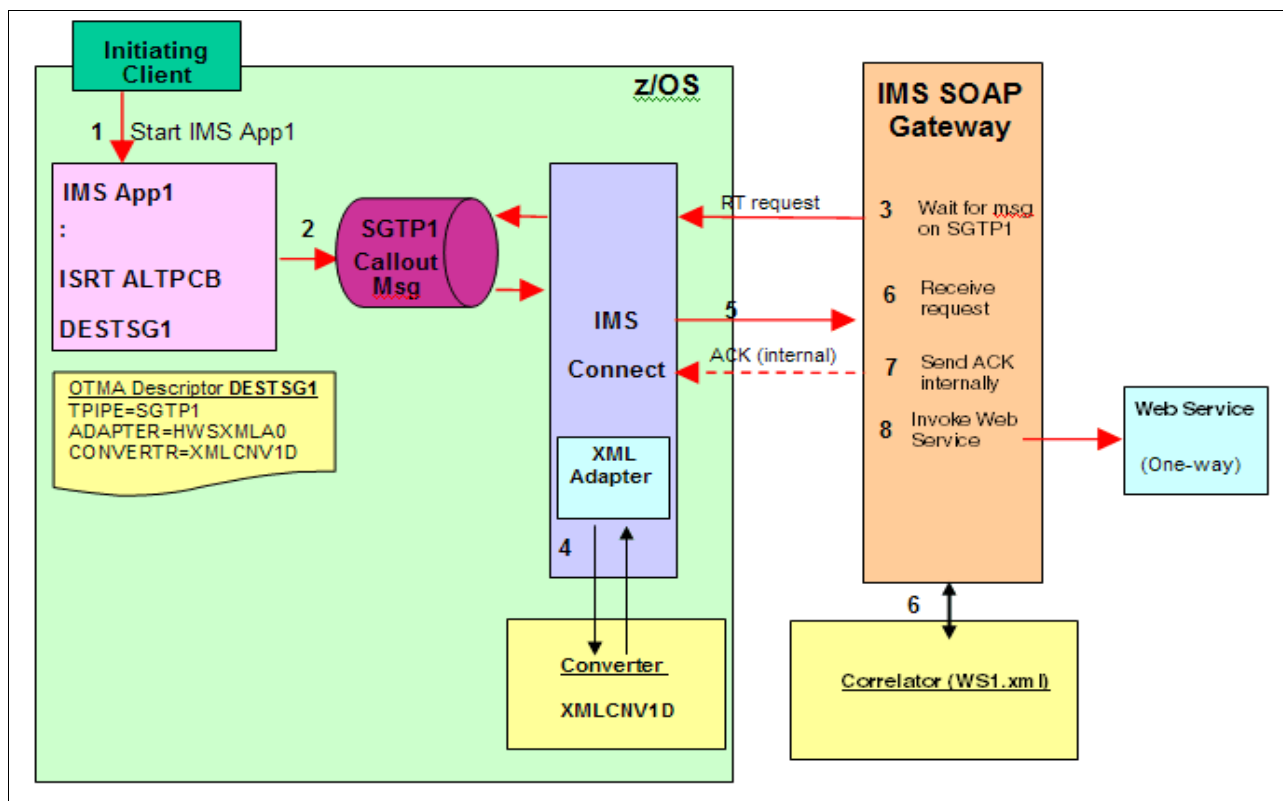


Figure 5-20 SOAP callout to a one-way web service invocation with no response

Here are the stages that are associated with this type of asynchronous callout flow:

1. A client starts IMS App1.
2. A one-way callout message is inserted into the ALTPCB by the IMS application App1 using the destination routing descriptor DESTSG1. The descriptor contains the TMEBER name, TPIPE name, XML Adapter name, and the associated Converter name.

As part of the destination routing descriptor processing, OTMA reads the descriptor DESTSG1, which contains a TPIPE name of SGTP1, and sends the callout message to the asynchronous hold queue SGTP1 TPIPE. The adapter name (HWSXMLA0) and converter name (XMLCNV1D) that are specified in the DESTSG1 descriptor are copied to the OTMA headers of the callout message. The callout message looks like Figure 5-21 after descriptor processing.

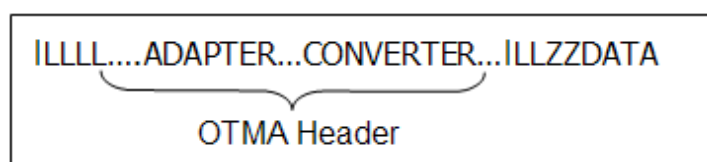


Figure 5-21 Callout message after OMTA descriptor processing

3. IMS SOAP Gateway establishes a shareable persistent TCP/IP connection with IMS Connect and sends a **RESUME TPIPE** using an option of 'No Auto' to retrieve messages from the SGTP1 TPIPE. The TPIPE is obtained from the connection bundle file when IMS SOAP Gateway starts.
4. IMS Connect retrieves the existing callout message from SGTP1 as part of the **RESUME TPIPE** request processing for IMS SOAP Gateway. It reads the OTMA headers for the adapter name and converter name. It calls the adapter HWSXMLA0 and passes the converter name XMLCNV1D to it.

After the activity completes, the XML Adapter calls the converter XMLCNV1D to perform the data transformation from COBOL application data to XML. The converter adds the service data prefix to the message so that IMS SOAP Gateway can use/read the appropriate correlator file and direct the request to the appropriate web service operation.

5. After the converter converts the callout message successfully, XML Adapter wraps the callout message in a <XMLAdapterOutput> tag and then IMS Connect adds a 4-byte length LLLL at the beginning of the message and sends the callout request message back to IMS SOAP Gateway. The resulting callout message is shown in Figure 5-22.

```
LLLL<XMLAdapterOutput><IOGCallout><IOGServiceData><WSID>WS1</WSID>...</IOGServiceData><Req>DATA</Req></IOGCallout></XMLAdapterOutput>LLZZ*CSM OKY
```

Figure 5-22 Callout message with XMLAdapterOutput tag

6. After IMS SOAP Gateway receives the callout request message, it reads the LLLL portion of the message and determines the length of the message to be read. It also parses the message to retrieve the WSID from the service data prefix. Based on the WSID value "WS1", it loads the corresponding correlator file, which is WS1.xml, and retrieves the web service information and properties for starting the web service. The request message for the web service is built.
7. After the callout message is processed successfully and the request message for the web service is built, IMS SOAP Gateway internally sends a positive acknowledgement (ACK) to IMS Connect such that the callout message can be removed from the corresponding TPIPE.
8. IMS SOAP Gateway sends the one-way callout request message to the web service using SOAP/HTTP. The callout message using SOAP/HTTP is shown in Figure 5-23.

```
<SOAP><Req>DATA</Req></SOAP>
```

Figure 5-23 Callout message with SOAP tag

Scenario 2: Callout to as one-way operation to a web service with a response returned to IMS by starting a second web service on IMS SOAP Gateway

In this scenario, presented in Figure 5-24 on page 203, IMS application App1 calls out as a one-way operation to Web Service 1, which starts another one-way operation on Web Service 2.

The first seven steps are similar to the previous scenario in that they provide details about how a web service is started. The additional steps show the details of a web service starting another pre-deployed web service, which calls an existing IMS application.

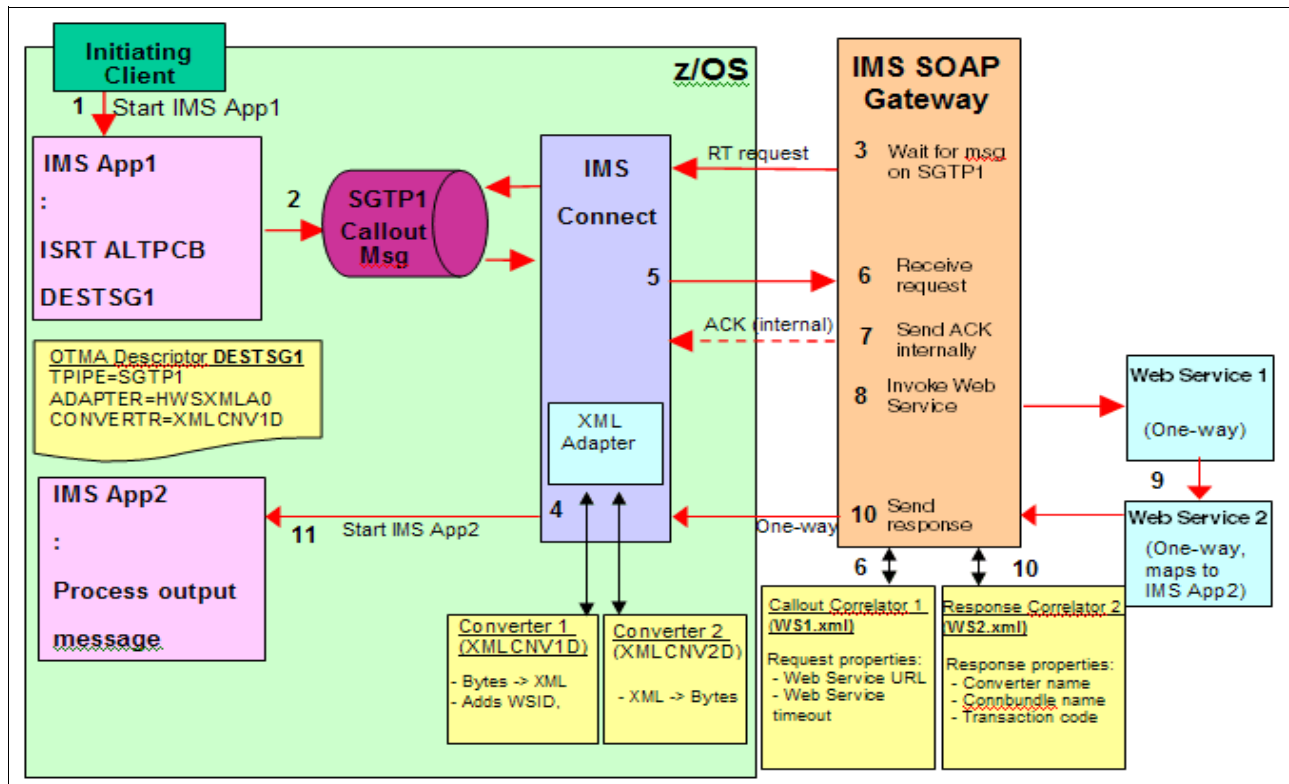


Figure 5-24 SOAP callout to a one-way web service invocation with a response

Complete the following steps:

1. A client starts IMS application App1.
2. A one-way callout message is inserted into the ALTPCB by the IMS application App1 using the destination routing descriptor DESTSG1. The descriptor contains the TMBER name, TPIPE name, XML Adapter name, and the associated Converter name.

As part of the destination routing descriptor processing, OTMA reads descriptor DESTSG1, which contains a TPIPE name of SGTP1, and sends the callout message to the asynchronous hold queue SGTP1 TPIPE. The adapter name (HWSXMLA0) and converter name (XMLCNV1D) that are specified in the DESTSG1 descriptor are copied to the OTMA headers of the callout message. The callout message looks like Figure 5-21 on page 201 after descriptor processing.

3. IMS SOAP Gateway establishes a shareable persistent TCP/IP connection with IMS Connect and sends a **RESUME TPIPE** to retrieve messages from the SGTP1 TPIPE. The TPIPE is obtained from the connection bundle file when IMS SOAP Gateway starts.
4. IMS Connect retrieves the existing callout message from SGTP1 as part of the **RESUME TPIPE** request processing for IMS SOAP Gateway. It reads the OTMA headers for the adapter name and converter name. It calls the adapter HWSXMLA0 and passes the converter name XMLCNV1D to it.

The XML Adapter calls the converter XMLCNV1D to perform the data transformation from COBOL application data to XML. The converter adds the WSID and the operation name information to the message so that IMS SOAP Gateway can use/read the appropriate correlator file and direct the request to the appropriate web service operation.

5. After the converter converts the callout message, the XML Adapter wraps the callout message in a <XMLAdapterOutput> tag and then IMS Connect adds a 4-byte length LLLL at the beginning of the message and sends the callout request message back to IMS SOAP Gateway. The resulting callout message looks like Figure 5-22 on page 202.
6. After IMS SOAP Gateway receives the callout request message, it reads the LLLL portion of the message and determines the length of the message to be read. It also parses the message to retrieve the service data. Based on the WSID value “WS1” from the service data, it loads the corresponding correlator file, which is WS1.xml, and retrieves the web service information and properties for starting the web service. The request message for the web service is built.
7. After the callout message is processed and the request message for the web service is built, IMS SOAP Gateway internally sends a positive acknowledgement (ACK) to IMS Connect, such that the callout message can be removed from the corresponding TPIPE.
8. IMS SOAP Gateway sends the one-way callout request message to Web Service 1 using SOAP/HTTP. No response is expected from Web Service 1. The callout message using SOAP/HTTP appears as shown in Figure 5-23 on page 202.
9. Web Service 1 runs, generates output, and calls Web Service 2. It is the responsibility of Web Service 1 to create and transform output in such a manner that it maps to the input request of Web Service 2.
10. IMS SOAP Gateway receives the Web Service 2 request and it loads the correlator 2 (WS2.xml) to obtain the information for IMS Appl2 and sends the data to IMS Connect. In this scenario, the IMS transaction code is supplied by the Web Service 2.
11. IMS Connect receives the input message and loads converter 2 (XMLCNV2D) for XML to bytes conversion. Then, it starts the IMS application Appl2 to process the data.

Scenario 3: Callout to a request-response web service

In this scenario, presented in Figure 5-25 on page 205, the IMS application calls out to a request-response operation on the web service, and a response is sent back to IMS SOAP Gateway, which starts a new IMS application (IMS application App2) for processing the output.

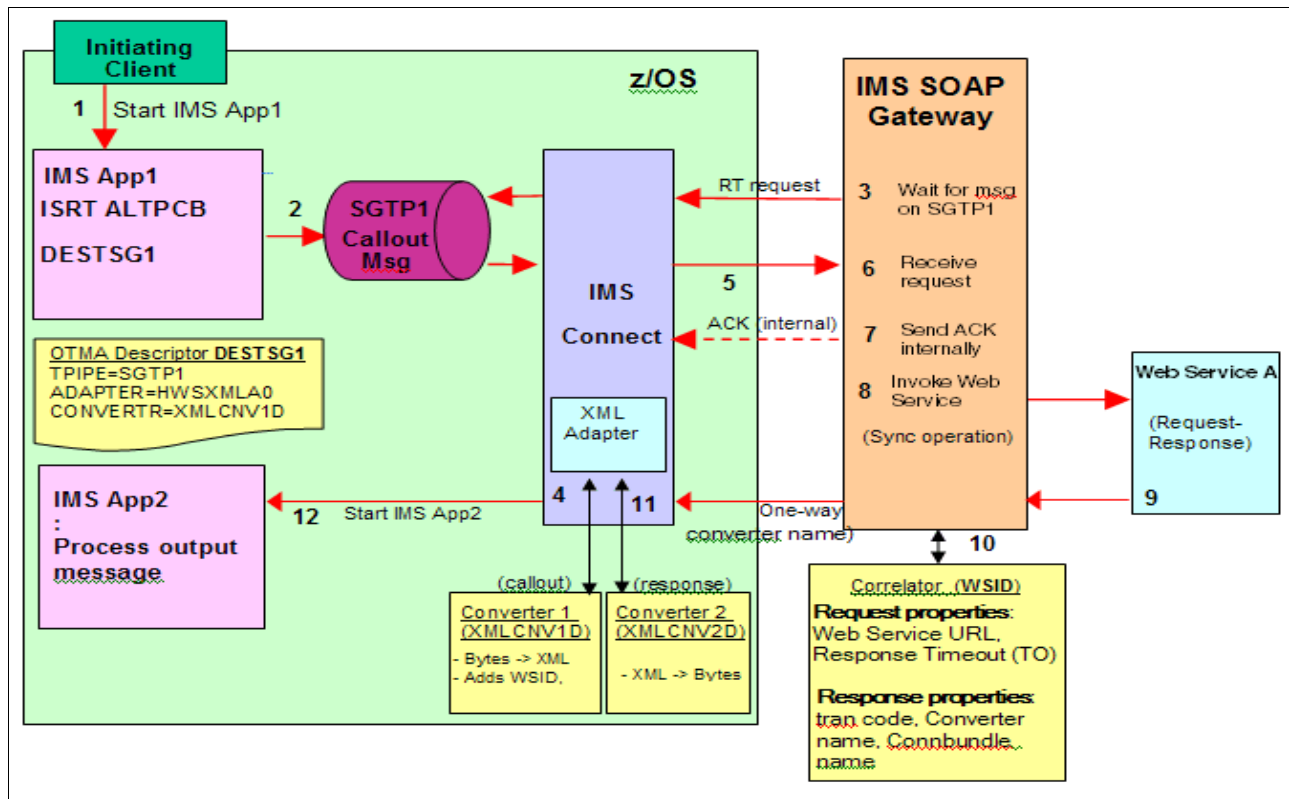


Figure 5-25 Message flow of SOAP callout to a request-response web service invocation

Complete the following steps:

1. A client starts IMS application App1.
2. A one-way callout message is inserted into the ALTPCB by IMS application App1 using the destination routing descriptor DESTSG1. The descriptor contains the TMBER name, TPIPE name, XML Adapter name, and the associated Converter name. The callout message looks like Figure 5-22 on page 202 after descriptor processing.

As part of the destination routing descriptor processing, OTMA reads the descriptor DESTSG1, which has a TPIPE name of SGTP1, and sends the callout message to the asynchronous hold queue SGTP1 TPIPE. The adapter name (HWSXMLA0) and converter name (XMLCNV1D) that are specified in the DESTSG1 descriptor are copied to the OTMA headers of the callout message.

3. IMS SOAP Gateway establishes a shareable persistent TCP/IP connection with IMS Connect and sends a **RESUME TPIPE** using an option of 'No Auto' to retrieve messages from the SGTP1 TPIPE. The TPIPE is obtained from the connection bundle file when IMS SOAP Gateway starts.
4. IMS Connect retrieves the existing callout message from SGTP1 as part of the RESUME TPIPE request processing for IMS SOAP Gateway. It reads the OTMA headers for the adapter name and converter name. It calls the adapter HWSXMLA0 and passes the converter name XMLCNV1D to it.

The XML Adapter calls the converter XMLCNV1D to perform the data transformation from COBOL application data to XML. The converter adds the WSID and the operation name information to the message so that IMS SOAP Gateway can use/read the appropriate correlator file and direct the request to the appropriate web service operation.

5. After the converter converts the callout message successfully, XML Adapter wraps the callout message in an <XMLAdapterOutput> tag and then IMS Connect adds a 4-byte length LLLL at the beginning of the message and sends the callout request message back to IMS SOAP Gateway. The resulting callout message appears as shown in Figure 5-22 on page 202.
6. After IMS SOAP Gateway receives the callout request message, it reads the LLLL portion of the message and determines the length of the message to be read. It also parses the message to retrieve the service data prefix. Based on the WSID value "WS1" from the service data prefix, it loads the corresponding correlator file, which is WS1.xml, and retrieves the web service information and properties for starting the web service. The request message for web service is built.
7. After the callout message is processed and the request message for the web service is built, IMS SOAP Gateway internally sends a positive acknowledgement (ACK) to IMS Connect such that the callout message can be removed from the corresponding TPIPE.
8. IMS SOAP Gateway sends the request to Web Service 1 using SOAP/HTTP and waits for the response synchronously on the same HTTP connection. The callout message using SOAP/HTTP appears as shown in Figure 5-23 on page 202.
9. Web Service 1 runs and generates an output response and sends it back to IMS SOAP Gateway.
10. IMS SOAP Gateway adds the transaction code App2, the adapter name, HWSXMLA0, and the converter name, XMLCNV2D (all retrieved from the correlator file in step 6) to the IRM and sends the response message to IMS Connect.
11. IMS Connect receives the response message and loads the response converter XMLCNV2D for XML to bytes conversion. IMS Connect takes the transaction code value and adds it to the appropriate place (LLZZTRCDDATA) in the message.
12. IMS starts the IMS application App2 to process the output data.

Rational Developer for System z can be used to import the SOAP Client WSDL, generate the correlator file, and create the XML Converter routine to be called by IMS Connect to do the XML transformation for the callout message and the COBOL copy member that represents the output message.

Considerations for callout usage

When you use the callout function, to restart or gracefully shut down the IMS SOAP Gateway server, complete the following steps:

1. Stop all the callout threads by using the IMS SOAP Gateway Management Utility. Give sufficient time for the threads to stop completely. The time that it takes to stop the threads is based on the callout poll interval property that is defined in the `imsssoap.properties` file. At a minimum, wait as long as the callout poll interval time.
2. Stop the IMS SOAP Gateway server.
3. When you are ready, start the IMS SOAP Gateway server. If the callout function is not disabled, the callout threads start automatically.

5.6 Synchronous callout with IMS SOAP Gateway

IMS SOAP Gateway V10.1 with the latest interim fix supports synchronous callout. With this new feature, your IMS applications are enabled to synchronously start external applications

With synchronous callout support, IMS applications call out synchronously and wait for the response to come back. Using this programming style, the IMS application program can initiate direct communication with other external application programs and receive the response in the same IMS transaction instance.

5.6.1 Synchronous callout support with IMS SOAP Gateway

Figure 5-26 is an overview of IMS Synchronous Callout support.

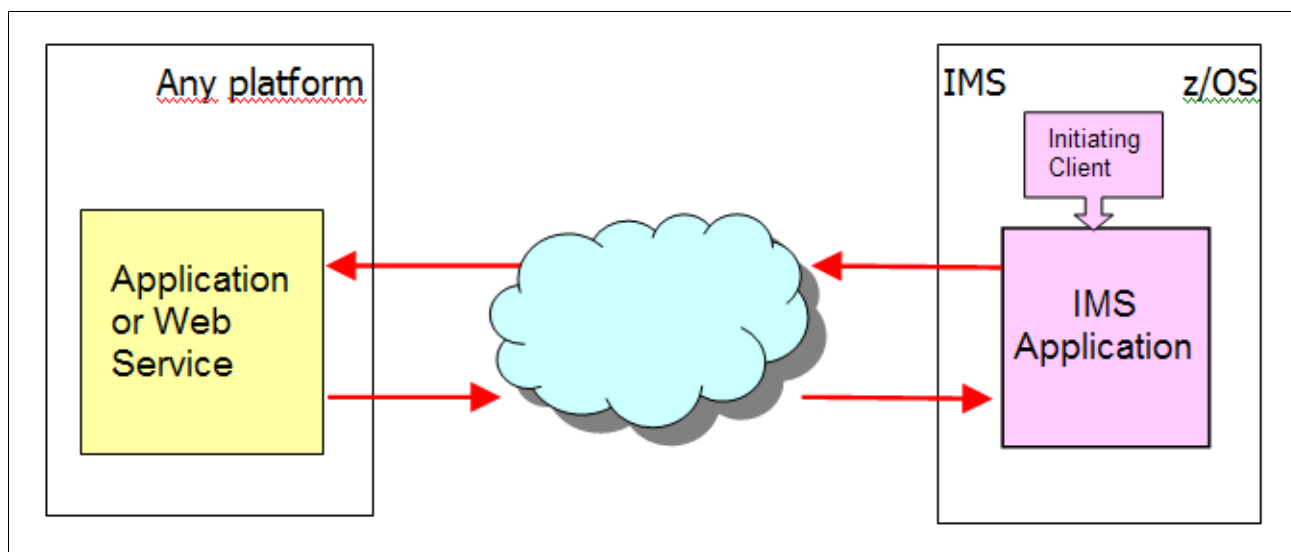


Figure 5-26 IMS Synchronous Callout support

Here are the IMS Synchronous Callout support enhancements:

- ▶ A new DL/I call, the ICAL SENDRECV call, for an IMS application to send out a synchronous callout request and receive the response back. The ICAL synchronous call function introduces a new IMS application programming style. The traditional IMS application programming style is an asynchronous programming model that is supported by the IMS message queue.
- ▶ The ICAL synchronous call function is supported only in the AIBTDLI interface in IMS Transaction Manager (TM) runtime environments for IFP, MPP, BMP, JMP, and JBP regions. ICAL is used to support IMS Synchronous Callout.
- ▶ Synchronous invocation to web services.
- ▶ Communication topology to enable an external application to process the callout request and to return the response to the same IMS application program instance. The IMS-dependent region is in a wait state when waiting for the response.
- ▶ Timeout capability to allow the IMS application to specify how long it can wait for the response message to come back, which can help to terminate the callout request and free the dependent region if the external service does not respond in a timely manner.

- ▶ Enhanced OTMA Resume TPIPE and Send-Only message processing functions for external application to pull synchronous callout request messages and return the response back.
- ▶ Relief of the 32 KB segmentation limitation for synchronous callout messages because the IMS message queue is not used. The maximum XML message size for both the request and response for synchronous callout messages is 32 MB for COBOL applications and 16 MB for PL/I applications, which enable IMS applications to send out and receive large messages.
- ▶ Concurrent processing with a correlation mechanism to allow callout requests to be sent out and response messages to be received in different connections and threads.
- ▶ Updated IMS commands to view synchronous callout request status.

Figure 5-27 shows an overview of the IMS Synchronous Callout flow.

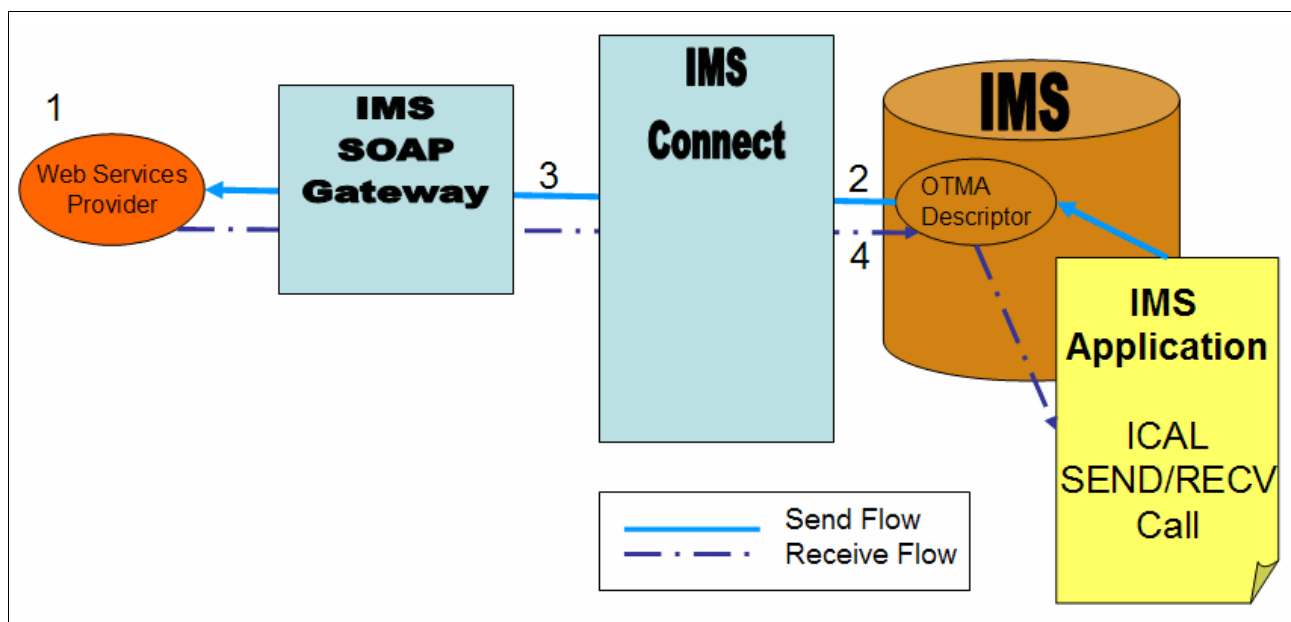


Figure 5-27 Overview of the IMS Synchronous Callout flow

Here is the synchronous callout flow:

1. The external application or server does not listen for callout messages because of the usage of the IMS OTMA Resume TPIPE function.
2. The IMS application program issues an ICAL SENDRECV synchronous callout request through an OTMA descriptor name, which defines the destination for the call.
3. The OTMA descriptor allows IMS to route the callout request through IMS Connect to the outbound destination, for example, a EJB/MDB running in WebSphere Application Server using the IMS TM Resource Adapter, a RYO IMS Connect TCP/IP application, or a web service provider through IMS SOAP Gateway.
4. After the callout request is processed, the response data is returned to the same IMS transaction instance.

In summary, after the synchronous callout request message is received, IMS SOAP Gateway processes the request, finds the web service identifier in the callout request message, and matches it with the deployed web services to locate and start the web service. The started web service, after processing, sends the callout response message back to the IMS application that is waiting synchronously for a response.

Starting the web service operation

For IMS Synchronous Callout, only one type of web service operation is supported. It is the request-response invocation, where a response message is returned synchronously back to the caller. IMS SOAP Gateway waits for the response data to come back synchronously from the same SOAP/HTTP connection.

The invocation of the web service is performed by a worker thread. Any errors that occur in the worker thread during the invocation of the web service is sent back with an error response to IMS Connect which, in turn, passes the error to the waiting IMS application.

5.6.2 Synchronous callout user scenarios

This section describes the various topologies where you can use synchronous callout.

Scenario 1: Multiple IMS applications

Figure 5-28 shows that multiple IMS applications can make callout requests concurrently. In this example, each IMS application uses a different descriptor for the target callout destination.

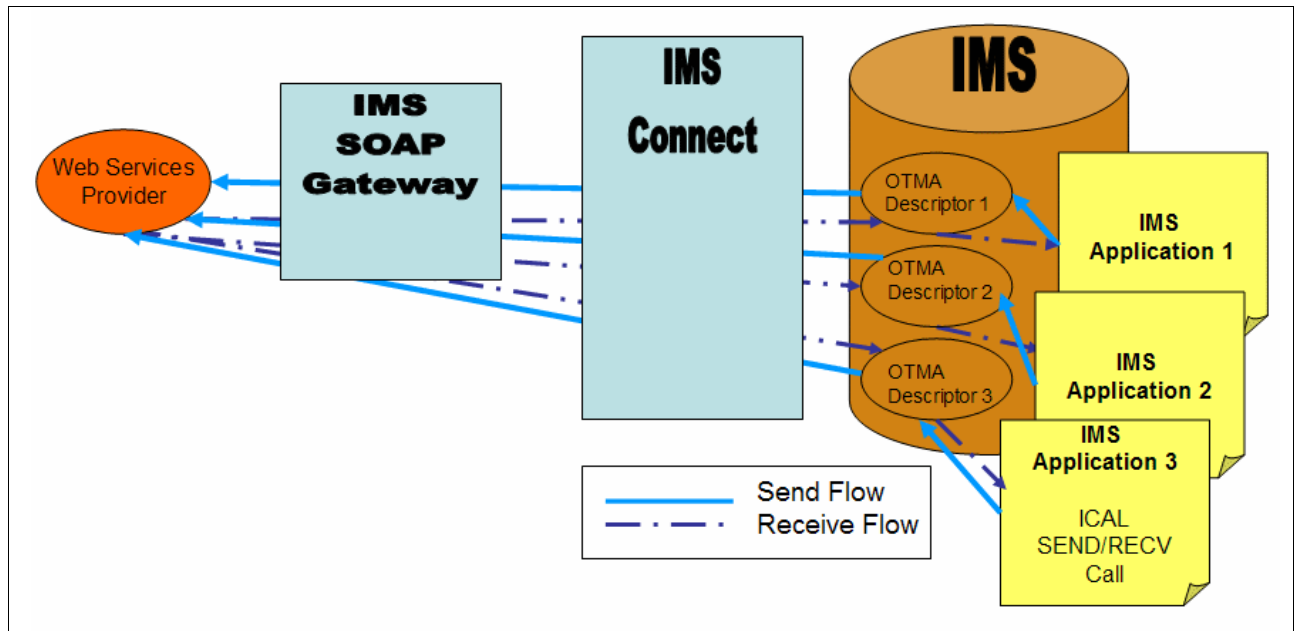


Figure 5-28 Callout flow with multiple IMS applications

Scenario 2: Multiple IMS Connect instances

Figure 5-29 shows a scenario where different IMS Connect instances are used to send the request and to receive the response messages. The correlation token that is sent inside the request and response message allows IMS to ensure that the response message goes to the same IMS transaction instance that initiates the synchronous callout request.

This scenario is common if Sysplex Distributor is configured and used in front of a number of IMS Connects. In this scenario, all the IMS Connects share a single external IP address that is managed by the Sysplex Distributor. When the response is returned, the Sysplex Distributor might route the response to any of the IMS Connect if the external server uses a different connection to return the callout response message.

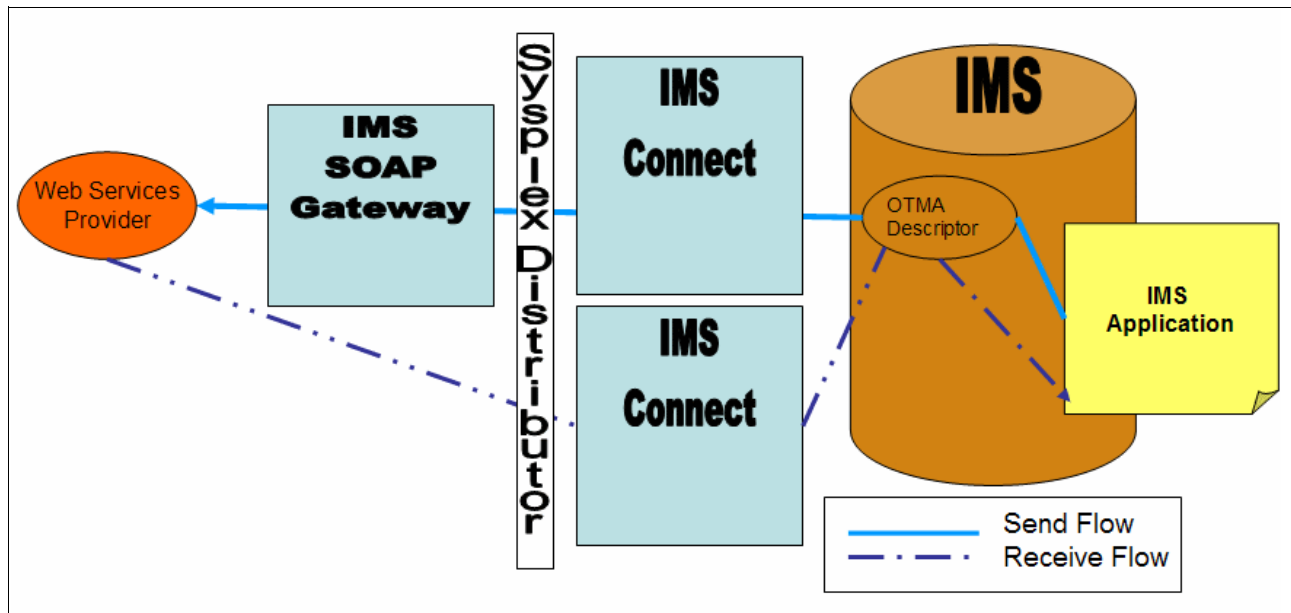


Figure 5-29 Callout flow with multiple IMS Connects

Scenario 3: IMS shared queues

Figure 5-30 on page 211 shows that IMS Synchronous Callout support can be used in a shared queue environment if the IMS is both the front-end and back-end system.

The Send Only response must be routed back to the same IMS that initiated the request; otherwise, the response message is rejected.

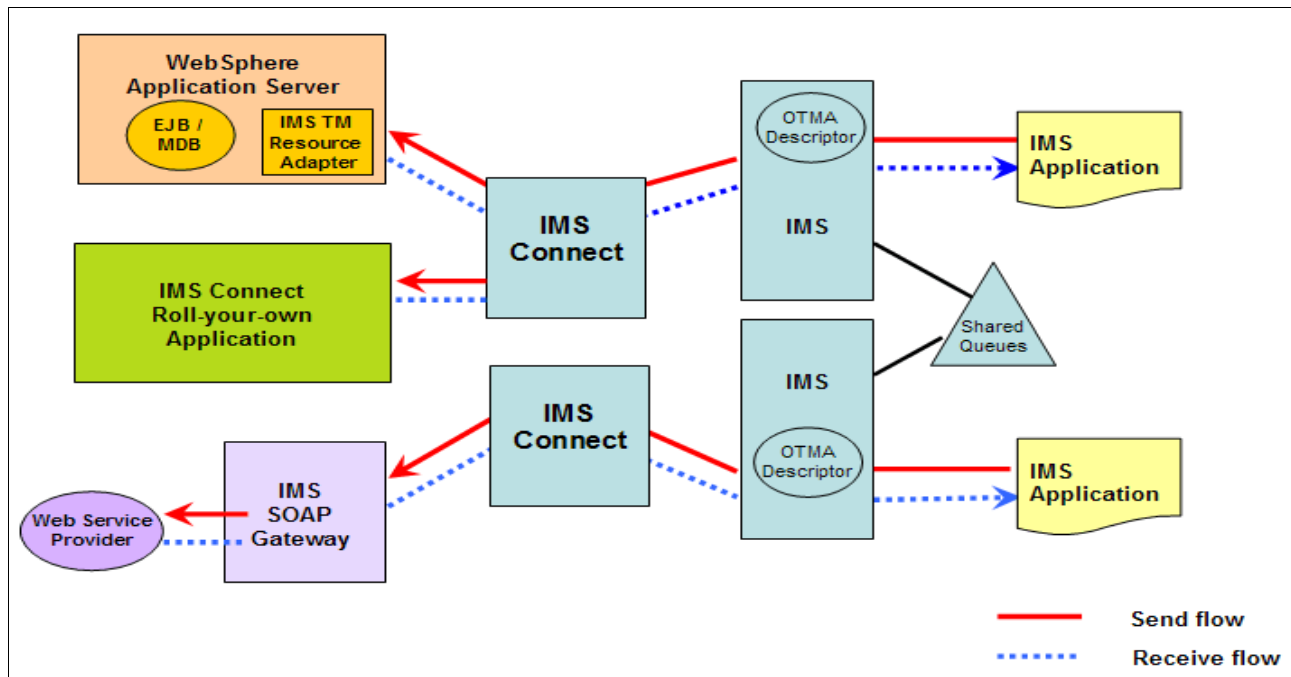


Figure 5-30 Callout flow with IMS shared queues

Scenario 4: Multiple servers for high availability

Figure 5-31 illustrates that multiple servers can be configured to achieve high availability and redundancy.

Server 1 can be a WebSphere Application Server, IMS SOAP Gateway, or RYO application. Server 2 can be connected to the same IMS Connect as Server 1 or a different IMS Connect instance. If IMS Connect instance attempts to deliver the callout request message to Server 1 and fails, IMS Connect sends the message back to OTMA, which allows the message to remain on the TPIPE and be retrieved by Server 2 to complete the processing.

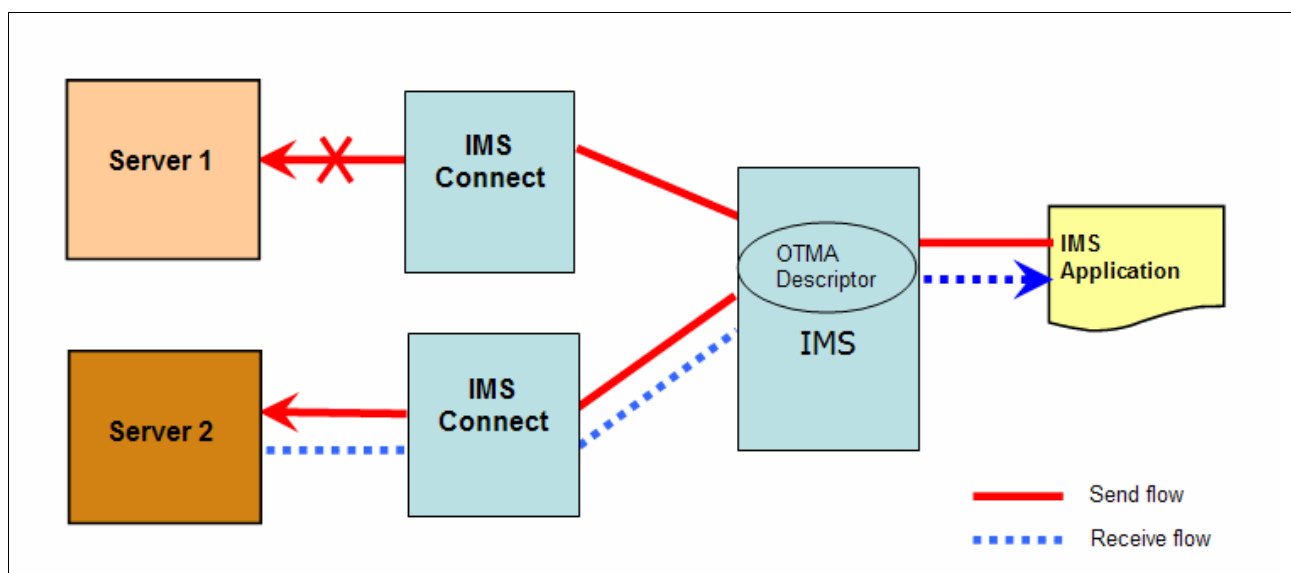


Figure 5-31 Callout flow with multiple servers

Requirements and restrictions for synchronous callout

This section provides the requirements and restrictions for synchronous callout.

Here are the requirements for synchronous callout:

- ▶ IMS SOAP Gateway Version 10.1 with the latest interim fix
- ▶ Rational Developer for System z Version 7.1.1.3 or later

Here are the restrictions for synchronous callout:

- ▶ Two-phase commit is not supported.
- ▶ Shared queue environments with different front-end and back-end IMS systems is not supported. The response message of a synchronous callout request must flow back to the originating IMS to be processed correctly.
- ▶ BMP or JBP applications running in a DBCTL environment are not supported.
- ▶ The callout processing inside the IMS application must be single-threaded, that is, the IMS application can only issue one synchronous callout call at a time. It must wait for the response (or timeout) before it can issue the next callout request.
- ▶ The maximum XML message size for both the request and response for synchronous callout messages is 32 MB for COBOL applications and 16 MB for PL/I applications.

5.7 Multisegment support

IMS SOAP Gateway V10.2 introduces multisegment support. The significance of this support is that IMS multisegment applications can now participate in SOA using IMS SOAP Gateway.

This support is provided for the XML adapter scenario in IMS Connect and IMS SOAP Gateway, with tool support from Rational Developer for System z Version 7.5. Multisegment support is provided only for the “provider” (inbound, both request / response) scenario and not for the “consumer” (callout) scenario.

As with single segment handling, the bulk of the multisegment handling is performed by the Rational Developer for System z generated XML converters, which are deployed in IMS Connect. The IMS Connect XML Adapter function uses the XML converters for the XML-to-bytes conversion on the way in and bytes-to-XML conversion on the way out of IMS. This approach still applies for multisegment support. IMS SOAP Gateway uses the Rational Developer for System z tools to support the generation of web services artifacts for multisegment message processing programs (MPPs).

Multisegment handling in SOAP Gateway is achieved in two steps:

1. Artifact generation using Rational Developer for System z (Rational Developer for System z) Version 7.5
2. Runtime handling in IMS Connect by the XML Adapter and the generated XML converters

Here are the prerequisites for multisegment support:

- ▶ IMS Connect V10 SPE APAR PK69366
- ▶ IMS SOAP Gateway V10.2
- ▶ Tool support: Rational Developer for System z V7.5

5.8 Security enhancements

SSL and HTTPS security allow data to be transferred in a secure manner between the web service and IMS SOAP Gateway and between IMS SOAP Gateway and IMS Connect.

IMS SOAP Gateway processes a SOAP request by stripping off the SOAP headers and passing the XML payload to IMS Connect. This request can be sent in plain format or over an SSL connection to protect the message.

The security properties between IMS SOAP Gateway, IMS Connect, and IMS are specified in the connection bundle. The security properties that must be specified and that are passed to IMS Connect are User ID, Password, and Groupname.

In addition, there are SSL parameters that are required for the IMS SOAP Gateway to communicate with the IMS Connect through TCP/IP SSL Sockets. These parameters are listed in 5.2, “IMS SOAP Gateway overview” on page 170. To specify this data, you must open the Connection Bundle file, which can be accessed through the IMS SOAP Gateway Management Utility.

If RACF=Y is specified in the HWSCFGxx member, IMS Connect issues a RACF call to verify these parameters. If the call is not successful, an error message is returned to IMS SOAP Gateway. If the call is successful, the User ID and Groupname are passed to OTMA.

The IMSLSECX exit is also started if it exists, but this exit does not verify the password. Figure 5-32 maps out the message flow under security controls.

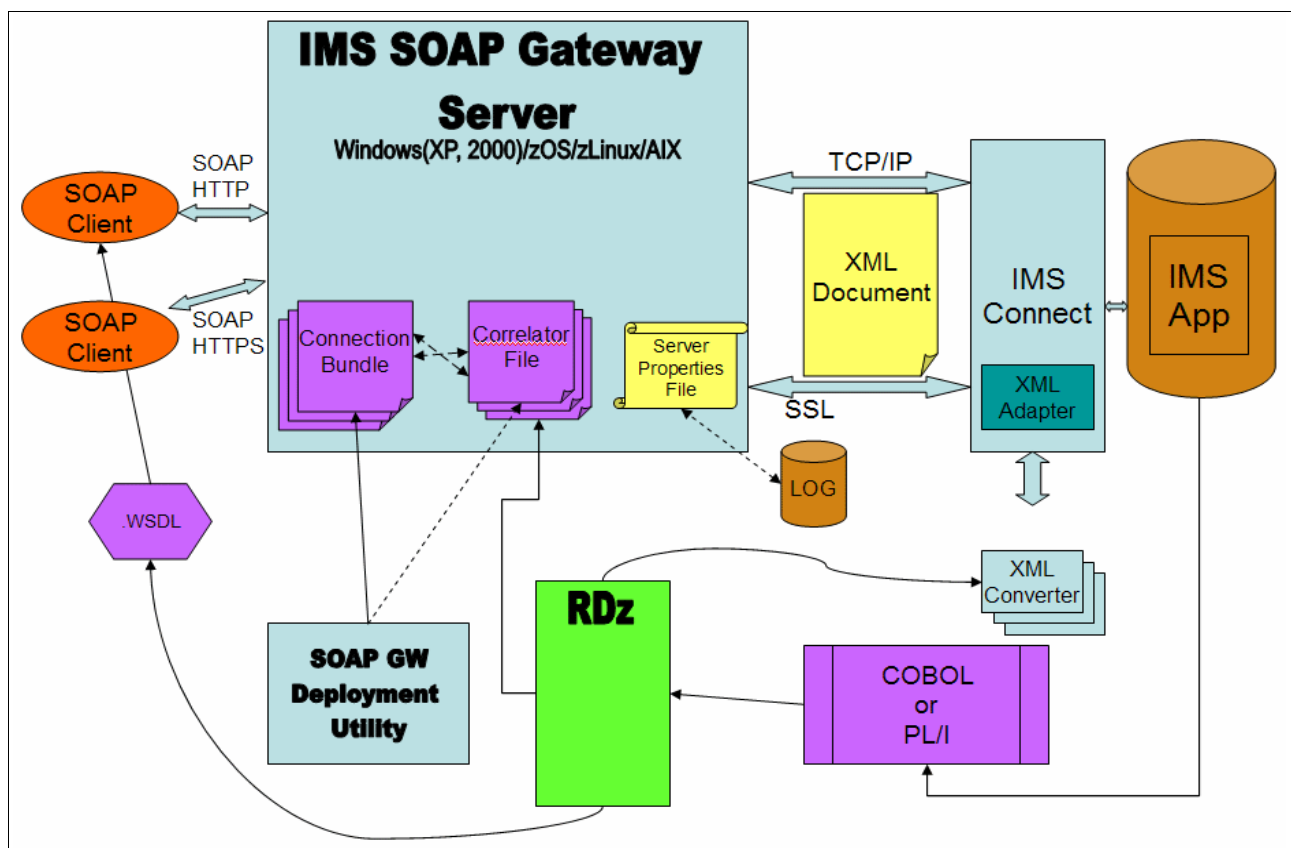


Figure 5-32 SSL/HTTP message flow

5.9 IMS SOAP Gateway features and compatibilities

Table 5-2 maps what features of IMS SOAP Gateway are compatible with which tools and versions of IMS and IMS Connect.

Table 5-2 Summary of IMS SOAP Gateway features and compatibilities

Feature	Minimum SOAP Gateway version	Minimum IMS version	Minimum IMS Connect version	Minimum Rational tool version
Accessibility Enablement	Version 9.2	Version 9	Version 9	Version 7.0
IMS Connect XML Adapter Support	Version 9.2	Version 9	Version 9	Version 7.0
Security Enhancements: SSL/HTTPS	Version 9.2	Version 9	Version 9	Version 7.0
Platform Support: z/OS and Linux for System z	Linux for System z: Version 9.2 z/OS: Version 10	Version 9 Version 10	Version 9 Version 10	Version 7.0 Version 7.1.1
Asynchronous Callout	Version 10.1	Version 10	Version 10	Version 7.1.1
Synchronous Callout	Version 10.1 +iFIX2	Version 10	Version 10	Version 7.1.1.3
Multiple Segment	Version 10.2	Version 10	Version 10	Version 7.5

For the current SOAP Gateway compatibility table, see the following website:

<http://www-01.ibm.com/support/docview.wss?uid=swg27036162>

Multiple segment support in IMS SOAP Gateway was announced in the IMS Version 11 QPP (Quality Partnership Program) announcement number 208-258, which can be found at the following website:

http://www.ibm.com/common/ssi/index.wss?DocURL=http://www.ibm.com/common/ssi/rep_ca/8/897/ENUS208-258/index.html&InfoType=AN&InfoSubType=CA&InfoDesc=Announcement+Letters&panelurl=index.wss%3Fbuttonpressed%3DNAV002PT090&paneltext=Announcement+letter+search



Java Platform, Enterprise Edition application technology

This chapter provides an overview of the Java Platform, Enterprise Edition and the tools that you can use with it, such as IBM Rational Developer for z and IBM WebSphere Application Server

Java Platform, Enterprise Edition provides containers for client applications, web components based on servlets and JavaServer Pages (JSP), and Enterprise beans (EJB) components. These containers provide deployment and runtime support for application components. They provide a federated view of the services that are provided by the underlying application server for the application components.

Containers can run on existing systems, for example, web servers for the web containers, and application servers, teleprocessing monitors, and database systems for EJB containers. These containers enable enterprises to use both the advantages of their existing systems and those advantages of Java Platform, Enterprise Edition.

This chapter covers the following topics:

- ▶ Java Platform, Enterprise Edition
- ▶ Key features of the IMS TM Resource Adapter
- ▶ Installing the IMS TM Resource Adapter
- ▶ Callin request support
- ▶ Callout request support

6.1 Java Platform, Enterprise Edition

Java Platform, Enterprise Edition (sometimes abbreviated as Java EE) is the Oracle enterprise Java computing platform. The platform provides an API and runtime environment for developing and running enterprise software, including network and web services, and other large-scale, multitiered, scalable, reliable, and secure network applications. Java EE extends the Java Platform, Standard Edition. Besides Enterprise Edition (EE), there is Standard Edition (SE) and Micro Edition (ME)

Enterprises can write, or rewrite, new applications using Java Platform, Enterprise Edition capabilities and can also encapsulate parts of existing applications in Enterprise beans, JavaServer Pages, or servlets. Enterprise applications access functions and data that is associated with applications running on Enterprise Information Systems (EIS).

Application servers extend their containers and support connectivity to heterogeneous EISs. Enterprise tools and Enterprise Application Integration (EAI) vendors add value by providing tools and frameworks to simplify the EIS integration task. For enterprise application integration, bidirectional connectivity between enterprise applications and EIS is essential. The Java Platform, Enterprise Edition Connector architecture defines standard contracts that allow bidirectional connectivity between enterprise applications and EISs. It also formalizes the relationships, interactions, and the packaging of the integration layer, thus enabling enterprise application integration.

The Java Runtime Environment (JRE) and Java Development Kit (JDK) are the actual files that are downloaded and installed on a computer to run or develop Java programs.

6.1.1 Developing Java applications

The Java development tools (JDT) project provides the tool plug-ins that implement a Java IDE supporting the development of any Java application, including Eclipse plug-ins. It adds a Java project nature and Java perspective to the Eclipse Workbench and a number of views, editors, wizards, builders, and code merging and refactoring tools. The JDT project allows Eclipse to be a development environment for itself. The Java visual editor is a tool that you can use to visually construct GUIs based on Swing Windows Toolkit (SWT) or Abstract Windows Toolkit (AWT). The Session Initiation Protocol (SIP) Tools feature provides a development environment for the creation of new SIP-based services. The feature enhances the existing Java Platform, Enterprise Edition development perspective to allow for the creation of SIP and converged HTTP/SIP applications. The enhanced Java EE perspective also includes support for the Servlet ARchive (SAR) archive format and includes a wizard for editing SIP deployment descriptors. You can bundle SAR archive files within a Java EE application archive, just like other Java EE components.

Java EE 5 provides simplified packaging rules for enterprise applications:

- ▶ Web applications use .WAR files.
- ▶ Resource adapters use .RAR files.
- ▶ Enterprise applications use .EAR files.
- ▶ The lib directory contains shared .JAR files.
- ▶ A .JAR file with Main-Class implies an application client.
- ▶ A .JAR file with @Stateless annotation implies an EJB application.
- ▶ Many simple applications no longer require deployment descriptors, including the following applications:
 - EJB applications (.JAR files)
 - Web applications that use JSP technology only

- Application clients
- Enterprise applications (.EAR files)

For more information about Java EE, see the official specifications:

- ▶ Java EE 5: JSR 244: Java Platform, Enterprise Edition 5 (Java EE 5) Specification
- ▶ Java EE 6: JSR 316: Java Platform, Enterprise Edition 6 (Java EE 6) Specification
- ▶ Java EE 7 JSR 342: Java Platform, Enterprise Edition 7 (Java EE 7) Specification

Here is a timeline of these Java versions:

- ▶ Java EE 5 (11 May, 2006)
- ▶ Java EE 6 (10 December, 2009)
- ▶ Java EE 7 (28 May, 2013)

Here are some Java EE 7 platform highlights.

The most important goal of the Java EE 7 platform is to simplify development by providing a common foundation for the various kinds of components in the Java EE platform. Developers benefit from productivity improvements with more annotations and less XML configuration, more Plain Old Java Objects (POJOs), and simplified packaging. The Java EE 7 platform includes the following new features:

- ▶ Batch applications for the Java platform.
- ▶ Concurrency utilities for Java EE.
- ▶ Java API for JSON processing (JSON-P).
- ▶ Java API for WebSocket.
- ▶ Enables developers to deliver HTML5 dynamic scalable applications.

If you are a beginner with Java, start with JUNO Eclipse Version. The JDT project provides the tool plug-ins that implement a Java IDE that supports the development of any Java application, including Eclipse plug-ins. It adds a Java project nature and Java perspective to the Eclipse Workbench and a number of views, editors, wizards, builders, and code merging and refactoring tools. The JDT project allows Eclipse to be a development environment for itself.

Web service development scenarios: Service flow projects

There are three possibilities to develop an application project: You can use a *bottom-up*, *meet-in-middle*, or *top-down* approach to develop web services in the context of developing a comprehensive web service that can collect and process data from multiple IMS applications, from other web services, or from both.

Service flow projects support these development approaches as follows:

- ▶ Bottom-up development

You use a service flow project in a bottom-up approach when your objective is to create a service from an existing application.

- ▶ Meet-in-middle development

You use a service flow project in a meet-in-middle approach when you already have both the WSDL file and the implementation component (application) and you need to create the additional support code that maps between the file and application.

- Top-down development

You use a service flow project in a top-down development approach when you already have the definition of a particular web service specified in a web service definition file (WSDL format) and you want to generate high-level language source files (in this case, COBOL source files) containing COBOL versions of the XML schema definitions that are specified in the WSDL file for the inbound message and the outbound message of the web service, and COBOL program modules that copy (“map”) the contents of the COBOL versions of the inbound message and outbound message to the XML schema definition versions of the inbound message and the outbound message, and vice versa.

These application development approaches are or embedded in to IBM Rational development products or in the Eclipse IDE

After you have installed the Eclipse Workbench IDE, you may start with the first Java projects HelloWorld and HelloWorld SWT (Figure 6-1) to discover the differences between SWT and non-SWT. You might find the cheat sheets in the IDE useful.

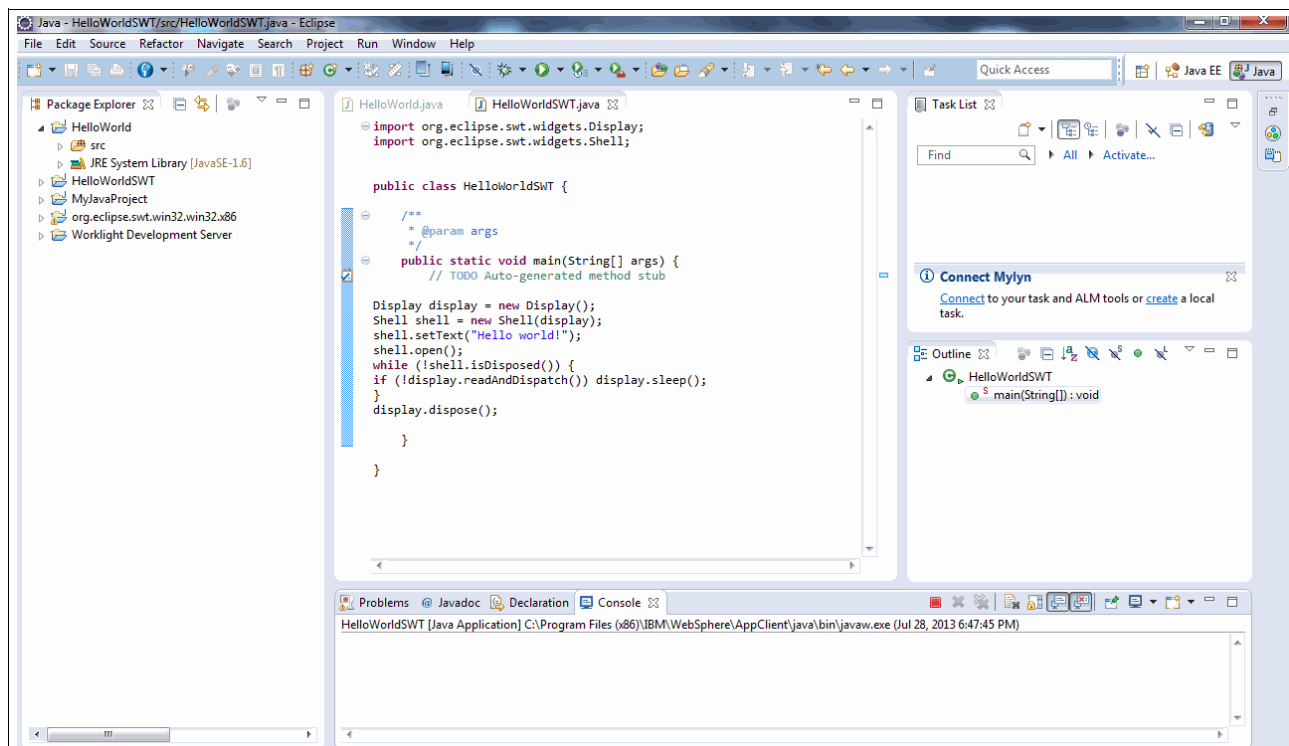


Figure 6-1 HelloWorld

Java Editor

The Java Editor example that is shown in Figure 6-2 on page 219 demonstrates the standard features that are available for custom text editors. It also shows how to register an editor for a file extension (in this case, .jav) and how to define a custom document provider for use by that editor. This example is only for demonstration purposes. Java editing support is provided by the Eclipse Java tools.

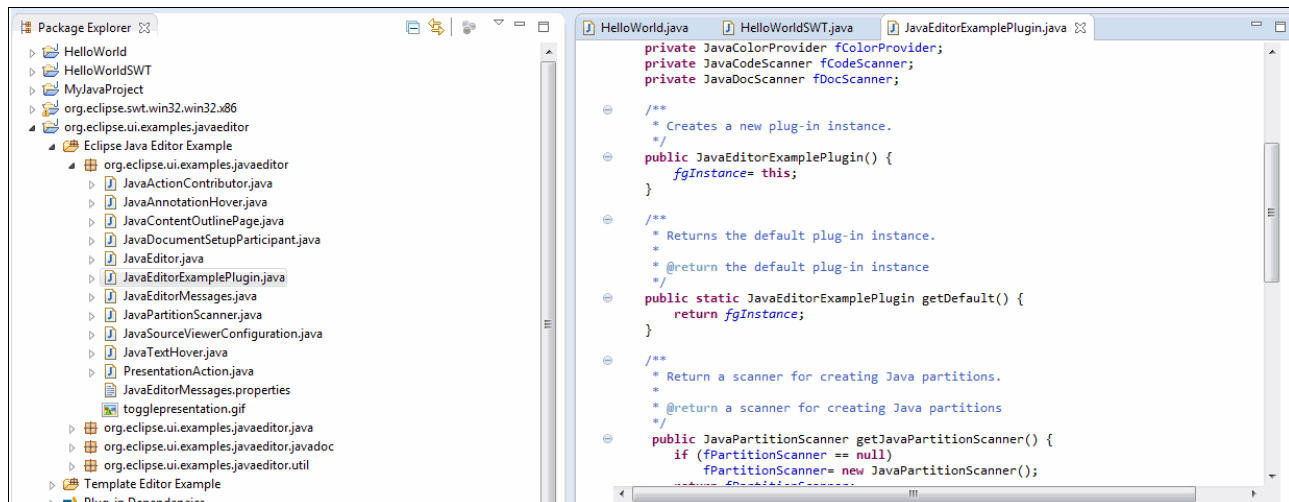


Figure 6-2 Java Editor example

What you can do with the sample

Browse the source code in the workspace. When you are ready, run the sample and follow the instructions in the help document. Later, you can rerun the sample by clicking the Run icon on the toolbar. If you place breakpoints in the code, you can debug it. Later on, you can debug the sample by clicking the Debug icon on the toolbar.

In the Help section, practice with the cheat sheets shown in Figure 6-3.

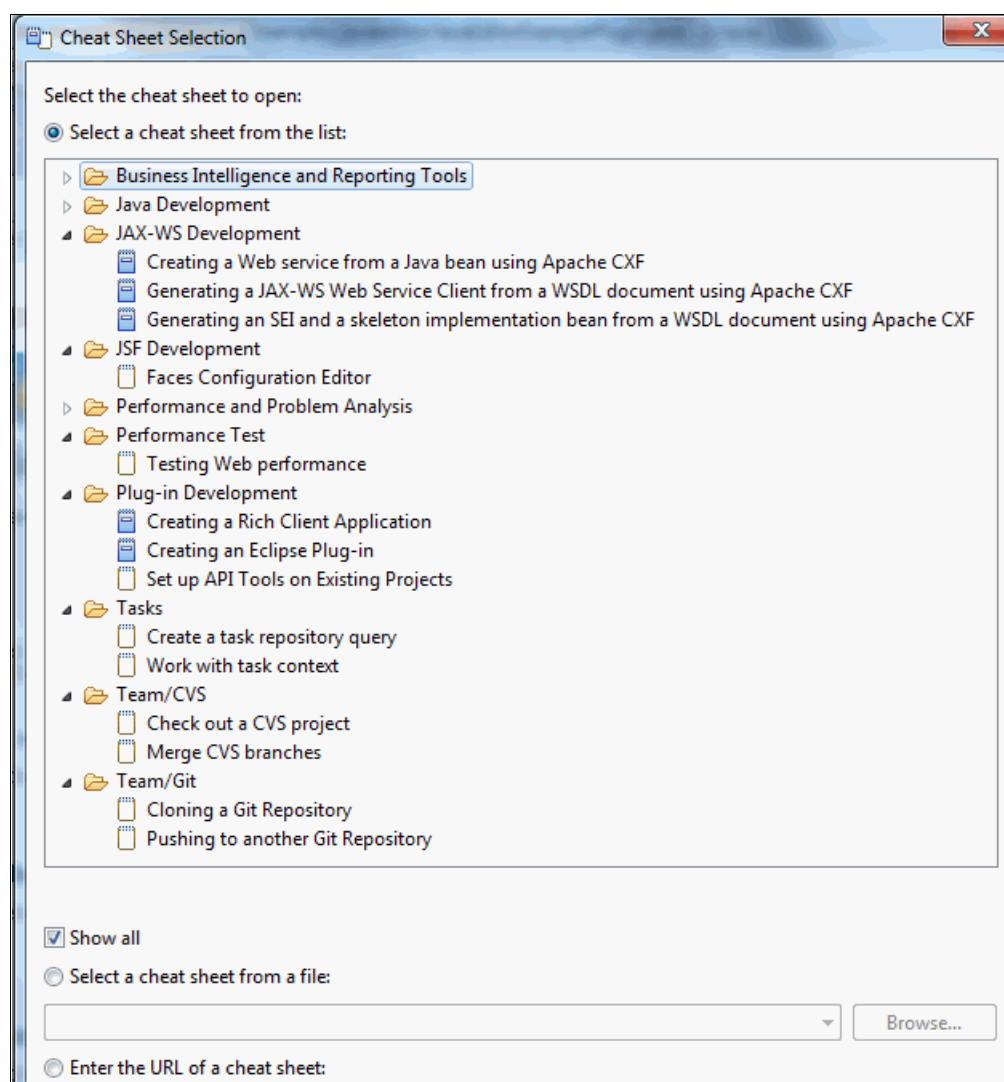


Figure 6-3 Cheat sheets

6.1.2 IBM Rational Developer for System z

IBM Rational Developer for System z consists of a mainframe system component and a workstation client component, as shown in Figure 6-4 on page 221.

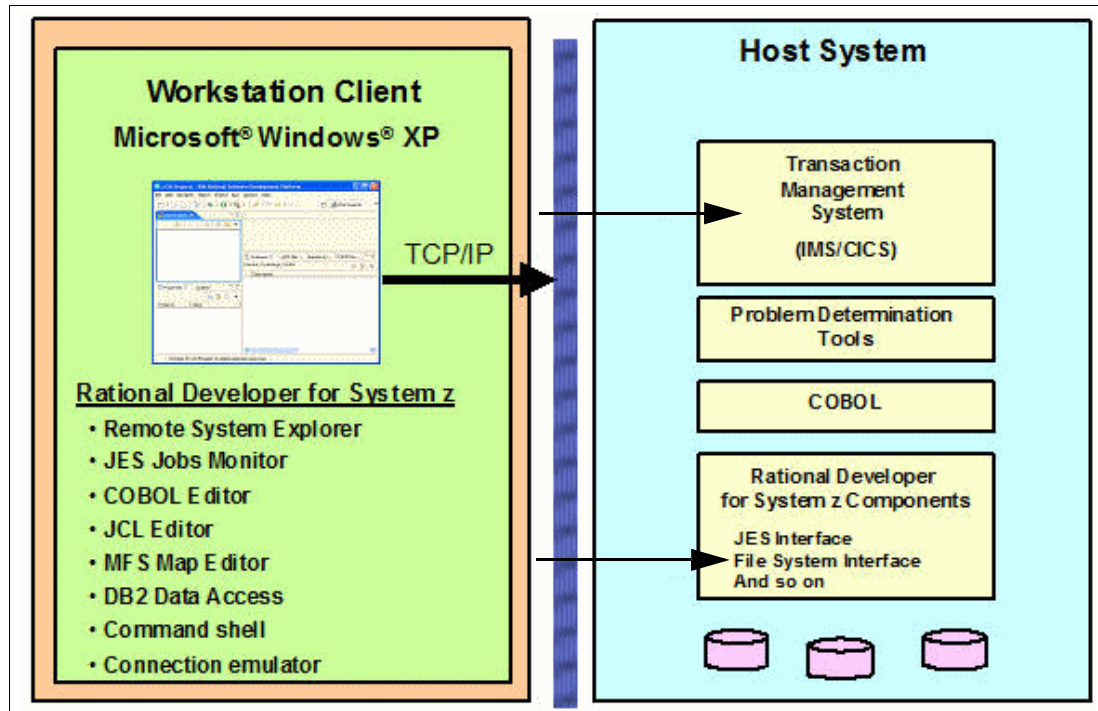


Figure 6-4 Rational Developer for System z

Mainframe system

The mainframe system component is also referred to as the remote system component. It is the part of the product that is installed on the System z mainframe. The remote system component is the System z mainframe to which you connect to access your data. This portion of the product is installed into a z/OS logical partition (LPAR) by the site system programmer.

The remote system component provides the interface between the workstation client and traditional development components, such as JES and the file system.

During setup, you create a connection between the workstation client and remote system components. The remote system component provides transaction management system functions to communicate with IMS and CICS. Problem determination tools, such as the IBM Debug Tool, are included.

Workstation client

The workstation client is the portion that is installed on a personal computer operating environment. It must be installed on each developer's computer. Throughout the product documentation, the term *IBM Rational Developer for System z* refers to the workstation component. The client software can be installed on a Microsoft Windows operating system or a Linux operating system. The two components communicate by using TCP/IP.

The user interface contains a set of development tools that support the development, maintenance, and web service enablement of enterprise applications.

- ▶ Access z/OS assets. You can organize, allocate, manage, and edit z/OS data sets and tools.
- ▶ Develop, maintain, compile, and assemble source files by using integrated language-sensitive editors for the following items: COBOL, PL/I, IBM High Level Assembler (HLASM), C/C++, Java, job control language (JCL), basic mapping support (BMS), and Message Format Service (MFS).

You can access the source directly from MVS data sets or through a number of source control management (SCM) systems.

- ▶ Manage and edit data, such as the following items:
 - Queued sequential access method (QSAM) or sequential data sets
 - Virtual Storage Access Method (VSAM) or indexed data sets
 - Relative and entry sequenced data sets
 - DB2 tables and views
 - Information Management System (IMS) databases
- ▶ Access to your z/OS Job Entry Subsystem (JES) to submit jobs and JES spool files to organize and manage batch jobs.

You can access other tools, such as wizards and declarative development tools that replace low-level coding, to create, deploy, and test DB2 stored procedures, web services, and Service Component Architecture (SCA), transform Unified Modeling Language (UML) models to COBOL, and use Enterprise Generation Language (EGL) to develop Web 2.0 and Java Platform, Enterprise Edition applications.

IMS snippets categories in Rational Developer for System z

Rational Developer for System z supports IMS categories among the categories in the Snippets view.

The categories for IMS snippets are as follows:

- ▶ IMS Transaction Management for COBOL
- ▶ IMS Database Management for COBOL
- ▶ IMS DB System Services for COBOL
- ▶ IMS Transaction Manager (TM) System Services for COBOL
- ▶ IMS Application Interface Masks for COBOL
- ▶ IMS DL/I Function Codes for COBOL

Management and services categories

You can use the first four categories to add IMS DL/I calls to your COBOL program. Each snippet in these categories corresponds to an individual DL/I function. Starting one of these snippets opens a dialog box that prompts you for details about the specific DL/I function call. At the top of the dialog box, you specify the interface type to use in the DL/I call from the list of interfaces that are compatible with the selected DL/I function call. The remaining fields correspond to the parameters of the corresponding DL/I call, with the following exceptions:

- ▶ When you use the language-independent interface (CEETDLI), you might need to complete the Type of Control Block field to specify whether you are using an AIB parameter.
- ▶ The CIMS, DPSB, GMSG, and INQY snippets have a field for you to complete in the subfunction code.

When you use these snippets with the System z LPEX Editor or COBOL Editor, references to nested fields of the AIB or PCB control blocks in the snippet code are replaced automatically with the actual fields that are present in the referenced AIB or control block. For example, when generating the line of code `MOVE LENGTH OF AIB TO AIBRLEN OF AIB`, the snippet generator looks up the field that is defined at offset 8 of the AIB control block that is selected in the dialog box and uses that as the name for the AIBRLEN field. If the AIB block cannot be parsed out of the code, a default value for the field is used.

When generating the DL/I call, the snippet code generator tries to look up a level 77 field whose VALUE clause equals the DL/I function and use that field in the call. For example, when creating the code for the GHN snippet, the generator parses the code and might find the declaration that is shown in Example 6-1.

Example 6-1 GET-HOLD-NEXT snippet

```
77 GET-HOLD-NEXT    PICTURE X(4)  VALUE 'GHN '.
```

GET-HOLD-NEXT is substituted in to the function call. If no match is found, the snippet name (GHN, in this case) is used in the function call.

These snippets add the DL/I function names as level 77 data structures.

IMS template generation

IMS PL/I top-down MPP template generation features separation of protocol-level logic from user-written business logic. The protocol-level logic is generated and used as is, and effectively shields the user-written business logic from the details of interacting with the IMS Message Queue and the IRZPWSIO API. Enhanced separation also allows for easier integration of existing business logic and increases the portability of new code for any future migrations.

Using the enhanced templates, IMS application programmers may immediately focus on writing or integrating business logic to process or create the input, output, or fault structures respective to each operation that is implemented by the MPP. Figure 6-5 illustrates how the separation of protocol-level logic from business logic is implemented in the enhanced templates.

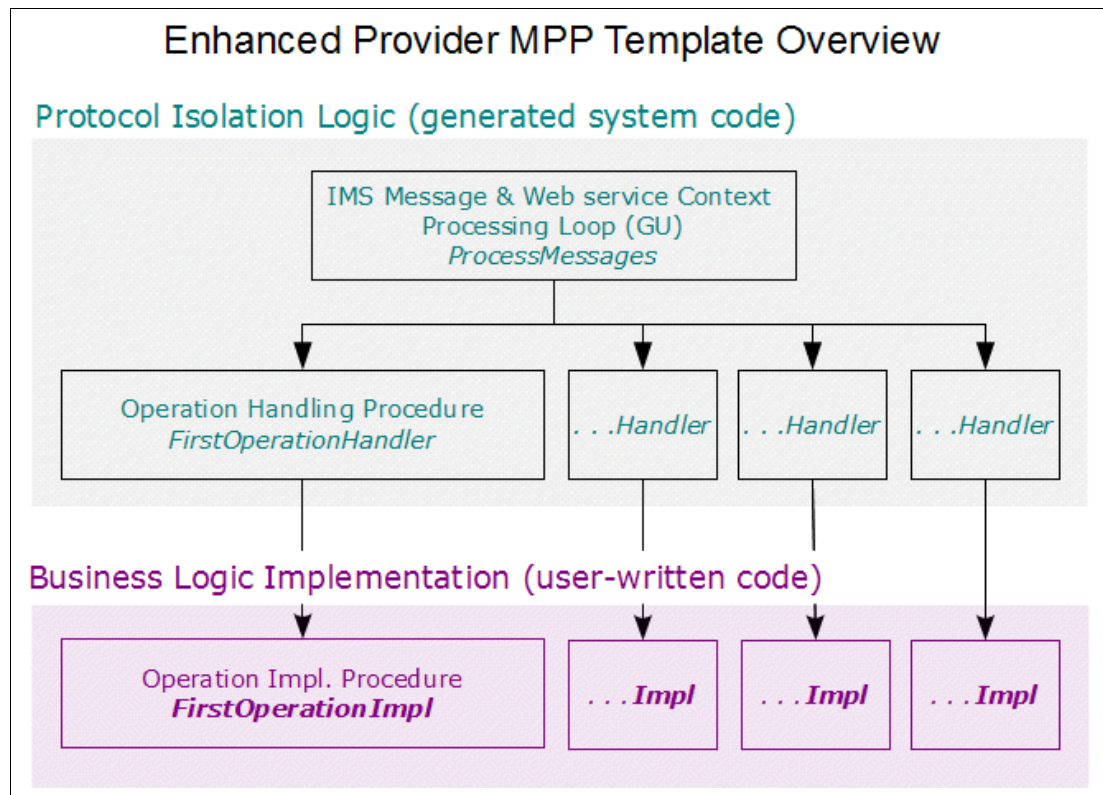


Figure 6-5 Enhanced Provider MPP template overview

As shown in Figure 6-5 on page 223, protocol-level logic exists in code-complete procedures that are named `OperationNameHandler`, which are dedicated to handling interactions with the IMS Message Queue and IRZPWSIO APIs on behalf of the respective user-written business logic in procedures named `OperationNameImpl`. All of the user-written business logic must be entered into the `OperationNameImpl` procedures. At run time, `OperationNameImpl` procedures are started by their respective `OperationNameHandler` procedure.

A.9, “Enhanced Provider MPP template sample” on page 451 shows an example of an enhanced template for a WSDL file that includes three operations:

- ▶ `getteam_1_0`
- ▶ `setteam_1_0`
- ▶ `ping_1_0`

The operations `getteam_1_0` and `setteam_1_0` might issue a SOAP fault.

6.1.3 The IBM IMS™ Resource Adapter

The IBM IMS™ Resource Adapter (IMS TMRA) (previously known as IMS Connector for Java (IC4J)) is used by Java applications, Java Platform, Enterprise Edition applications, or web services to access IMS transactions that are running on host IMS systems.

It is also used by IMS applications that run in IMS dependent regions to make asynchronous and synchronous (supported through IMS Version 10 maintenance and Version 11) callout requests to external web services.

In addition, the IMS TMRA implements the J2C Common Client Interface (CCI), which is a programming interface that allows your application to communicate with the IMS Transaction Manager.

The IMS TMRA is used with a Java EE server, such as IBM WebSphere Application Server, when Java applications (Java EE artifacts in the WebSphere Application Server) access an IMS transaction that is running on a host IMS system. The IMS TMRA also enables an IMS application to act as a client to start applications in a Java EE server.

Submitting commands to IMS

Although the IMS TMRA is intended primarily for you to run transactions on a host IMS system through a Java EE application, you can also issue IMS commands that are supported by IMS OTMA from your Java applications.

The IMS TMRA uses the host product, IMS Connect, to access IMS. IMS Connect uses the cross-system coupling facility (XCF) to access IMS through OTMA.

Only certain IMS commands can be submitted through the IMS OTMA interface. Because the IMS TMRA accesses IMS through OTMA, IMS commands that are supported by OTMA are the only commands that can be submitted to IMS by an application that uses the IMS TMRA.

The output of an IMS command is a message that consists of one or more segments of data. The output of some IMS commands is a DFS message. For example, the output of most `/START` commands is usually the message `DFS058I START COMMAND COMPLETED`. Other IMS commands do not return DFS messages. For example, `/DISPLAY` commands return multiple segments of data representing lines of display information.

To treat both types of output the same, you must set the `imsRequestType` property of the `IMSInteractionSpec` class to 2 (`IMS_REQUEST_TYPE_IMS_COMMAND`). This value indicates to the IMS TMRA that the interaction is an IMS command, and to treat DFS messages as normal output and not as Java exceptions.

Commands that can be submitted to IMS by an application that uses the IMS TMRA can be found in the IMS commands reference.

To have the full benefit of the adapter code, the adapter must be installed in the WebSphere Application Server and used from Java EE artifacts, but RYO Java client programs can also use the code.

6.2 Key features of the IMS TM Resource Adapter

Support for the IMS TMRA is based on Sun's Java EE Connector Architecture (JCA) 1.5. For IMS TMRA versions 10, 11, and 12, there is the following support:

- ▶ Supports development of applications that can submit transactions to IMS Transaction Manager through IMS Connect.
- ▶ Provides tool support for the development of Java EE applications, web services, and business processes that access IMS transactions in various Rational and WebSphere integrated development environments.
- ▶ Provides a JCA Resource Adapter (RAR) for deployment to the WebSphere Application Server and WebSphere Process Server runtime environment on many platforms, including z/OS and Linux on System z.
- ▶ Supports connection pooling and reuse.
- ▶ Supports global transaction processing and two-phase commit.
- ▶ Supports component-managed and container-managed security, including support for run-as thread identity.
- ▶ Supports SSL communication between the IMS TMRA and IMS Connect, including support for SSL null encryption.
- ▶ Supports most types of interactions with IMS application programs, including the retrieval of asynchronous output messages.
- ▶ Support for IMS applications to start Java EE applications asynchronously, and synchronously in callout mode.

IMS TM Resource Adapter components

There are two key components that make up IMS TMRA:

- ▶ A runtime component that must be deployed on the WebSphere Application Server. A deployed IMS TMRA in WebSphere Application Server fulfills all the rules of the JCA 1.5 requirements.
- ▶ A development component that lets you create your application by using an integrated development environment (IDE), such as IBM Rational Application Developer for WebSphere Software, IBM Rational Software Architect, IBM WebSphere Integration Developer, and IBM WebSphere Transformation Extender, as part of the optional J2C feature.

The J2C wizards in these IDEs enable you to create IMS connector code to be used from Enterprise JavaBeans (EJB) components, web services, or from stand-alone programs.

Supported platforms

The IMS TMRA runtime component supports WebSphere Application Server on the following platforms:

- ▶ IBM AIX®
- ▶ HP-UX
- ▶ Linux
- ▶ Linux for System z
- ▶ Solaris
- ▶ Windows
- ▶ z/OS and IBM OS/390®

6.2.1 Features that are introduced in IMS TM Resource Adapter Version 10.2

IMS TMRA Version 10.2 provides the following new features:

- ▶ Support for the invocation of IMS applications by using complex data types from distributed platforms

With the WebSphere Transformation Extender Design Studio, you can build applications that send complex data formats to and from IMS applications. WebSphere Transformation Extender is a transaction-oriented, data integration solution that automates the transformation of high-volume, complex transactions across the enterprise. You can use its Type Designer to generate a type tree from your COBOL copybook and then specify rules for transforming and routing the data in the Map Designer.

- ▶ Support for reestablishing any stale connection in a connection pool when the connection encounters a communication failure. Using this enhancement, you can recycle IMS Connect during system maintenance without resubmitting any IMS TMRA interactions from the client application.
- ▶ IMS MFS SOA support

IMS MFS SOA support is integrated with the Enterprise Metadata Discovery (EMD) framework in Rational Application Developer Version 7.5 or later to transform existing MFS-based IMS applications into J2C beans and J2C Java Data Binding classes. After you create a J2C bean, you can create Java EE artifacts, such as JSPs, EJBs, or web services, to deploy the generated J2C bean.

IMS MFS SOA support is supported in the following WebSphere Application Server V6.1 runtime environments:

- Windows
- z/OS

- ▶ Rerouting undeliverable output messages to a specified destination for commit mode 0, SYNC_SEND interactions on shareable persistent sockets

The reroute function allows undeliverable output messages to be rerouted to a specified destination. Previously, the reroute function was supported for only commit mode 0 (CM0), SYNC_SEND_RECEIVE interactions on shareable persistent sockets. This enhancement supports the reroute function on CM0, SYNC_SEND interactions on shareable persistent sockets.

6.2.2 Features that are introduced in IMS TM Resource Adapter Version 11

IMS TMRA Version 11 provides the following new features:

- Support for WebSphere Transformation Extender:

WebSphere Transformation Extender is the new name for the WebSphere DataStage® TX (formerly Ascential DataStage TX) product. WebSphere Transformation Extender performs transformation and routing of data from source systems to target systems in batch and real-time environments. The sources can include files, relational databases, message-oriented middleware (MOM), packaged applications, or other external sources. After retrieving the data from its sources, WebSphere Transformation Extender transforms it and routes it to any number of targets where it is needed, providing the appropriate content and format for each target system.

Figure 6-6 positions WebSphere Transformation Extender with respect to IMS TMRA.

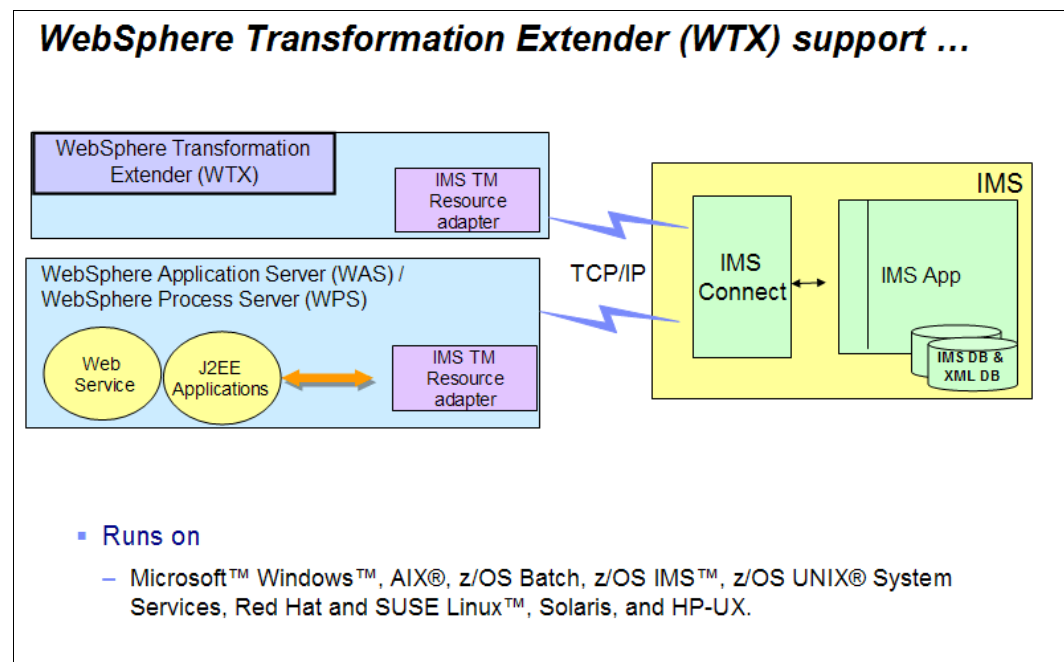


Figure 6-6 WebSphere Transformation Extender support

Here are the software requirements for WebSphere Transformation Extender:

- IMS and IMS Connect V10 or later
 - IMS TMRA V10.2 (APAR PK64663 provides this support for IMS TMRA V10.1.1)
 - WebSphere Transformation Extender V8.2.0.2
- IMS TMRA socket reconnect support

Socket reconnect support provides a retry and reconnect capability to re-establish a connection when encountering communication failures in sending / receiving and request / response through IMS Connect from a WebSphere Application Server, without user intervention. This retry function is implemented for Sync Level Confirm interactions to avoid duplicate interactions.

During system maintenance previously, customers would recycle IMS Connect, which caused WebSphere Application Server applications to fail because of persistent connections and connection pooling. This solution avoids unnecessary errors and quickly re-establishes connections, thus enhancing IMS availability and operational characteristics to allow these applications to seamlessly recover from connection errors.

- ▶ **IMS TMRA send only reroute support**
Send only reroute support reroutes current Sync-Send (Send-Only) interactions for CM0. With the reroute flag set to True and the reroute TPIPE name specified, any IOPCB output message resulting from the Send-Only transaction is queued to the reroute TPIPE.
- ▶ **Message Format Services (MFS) Business Process Execution Language (BPEL) support**
MFS BPEL support for the WebSphere Integration Developer assists integration developers with business-to-business transformation. This support creates web services/EJB/JSPs out of existing MFS-based IMS applications, and provides support for Service Component Architecture (SCA), Enterprise Metadata Discovery (EMD) 1.1, and BPEL. It includes a GUI wizard that allows you to parse the MFS source files and generate the service definition and artifacts. The generated application can be deployed to run on WebSphere Process Server for business orchestration and choreography.
- ▶ **Synchronous Callout support**
Synchronous callout (also available through an IMS Version 10 SPE) for IMS TMRA manages the correlation of a callout request. This enables IMS applications to start external applications and synchronously receive a response in the same IMS transaction instance. External applications and servers include Java EE applications (EJB/MDB).

6.2.3 New features in IMS TM Resource Adapter Version 12

IMS TMRA Version 12 adds support for WebSphere Application Server Version 8 and its resource workload routing function, support for multiple data stores per IMS activation specification for callout messages, and takes advantage of various enhancements in IMS.

Support for WebSphere Application Server V8 and its resource workload routing function

WebSphere Application Server V8 introduces a new resource workload routing function that offers data source and connection factory failover and subsequent failback from a predefined alternative or backup resource. This function enables applications to easily recover from resource outages, such as database failures, without having to embed alternative resource and configuration information.

You can specify two connection factories for the IMS TMRA, and then specify the second connection factory to be the alternative resource for the first connection factory.

Support for multiple data stores per IMS activation specification for callout messages

This enhancement enables a single message-driven bean (MDB) to pull callout messages from more than one IMS data store. For a shared-queues environment, this enhancement removes the requirement to duplicate the MDB application to connect to each IMS member in the shared queue.

Data store connection failure recovery for callout messages

When the back-end IMS is not available, in addition to the attempt to reconnect to IMS Connect, the resource adapter also attempts to reconnect to the IMS data store. A new -1 value is added to the IMSActivationSpec retryLimit property to support indefinite retry for connection to IMS Connect and the IMS data store. When the retry is successful and the connection is restored, informational messages are provided to indicate the successful reconnection.

IMSActivationSpec property configuration for message-driven beans

The values of the properties of the IMSActivationSpec object describe the inbound communication from IMS to be used by message-driven beans.

You must configure an instance of the IMSActivationSpec object when you deploy a message-drive bean in order for the IMS TMRA to communicate with IMS Connect for retrieving and responding to synchronous callout messages.

Table 6-1 describes the properties for the IMSActivationSpec object, which is the implementation of the J2C activation specification in IMS TMRA. The values for these properties are specified and configured by using the administrative console in WebSphere Application Server.

Table 6-1 Properties for the IMSActivationSpec object

Properties	Description
dataStoreName	The IMS data store name. The name must match the ID parameter of the Datastore statement that is specified in the IMS Connect configuration member when IMS Connect is installed. It also serves as the XCF member name for IMS during cross-system coupling facility (XCF) communications between IMS Connect and OTMA. With Version 12 and later, the data store name can be a comma-separated list of data store names, enabling one instance of the IMSActivationSpec class to pull callout messages from more than one IMS data store.
groupName	The security authorization facility (SAF) group name.
hostName	The IMS Connect host name.
password	The SAF password.
portNumber	The IMS Connect port number.
queueNames	A comma-delimited list of IMS OTMA tpie names for the callout (synchronous or asynchronous) messages. Queue names must be 1 - 8 alphanumeric characters (A-Z, 0-9, @, #, and \$).
retryInterval	The time delay in milliseconds before the IMS TMRA tries to check on the availability of the data store.
retryLimit	The maximum number of times the IMS TMRA attempts to reconnect to IMS Connect if a connection is lost because of IMS Connect or IMS data store availability issues. For Version 10.5.1, Version 11.3.1, and Version 12 and later, when the retryLimit value is set to -1, the IMS TMRA tries indefinitely to reconnect to IMS Connect if the back-end IMS Connect or IMS data store is restarted.
SSLEnabled	Instructs the IMS TMRA to create a Secure Sockets Layer (SSL) socket connection to IMS Connect by using the specified host name and port number in the IMSActivationSpec object. This property is valid for TCP/IP connections only.
SSLEncryptionType	Specifies the SSL encryption type. Valid values are strong and weak.
SSLKeyStoreName	The name, including the full file path, of the keystore for TCP/IP SSL communications. Private keys and their associated public key certificates are stored in password-protected databases called <i>keystores</i> . Trusted certificates can also be stored in the keystore, and the truststore property can either be empty or point to the keystore file. An example of a keystore name is c:\keystore\MyKeystore.ks. The file can have other file extensions.
SSLKeyStorePassword	The password for the keystore for TCP/IP SSL communications.

Properties	Description
SSLTrustStoreName	The name, including the full path, of the keystore file that contains security credentials (certificates) for TCP/IP SSL communications. A value for the SSLTrustStoreName property is not mandatory if a keystore is used. A truststore file is a key database file that contains public keys or certificates. Private keys can also be stored in the truststore, and the keystore property can either be empty or could point to the truststore file. An example of a truststore name is c:\keystore\MyTruststore.ks. The file can have other file extensions.
SSLTrustStorePassword	The password for the truststore for TCP/IP SSL communications.
userName	The SAF user name.

Properties in the J2C activation specification that are not listed in Table 6-1 on page 229 are not supported by the IMS TMRA.

Support for IMS V12 OTMA DFS2082 messages for commit-then-send CM0 transactions

With this IMS V12 enhancement, for CM0 transactions, if the IMS application does not reply to the IOPCB or complete a message switch to another transaction, OTMA issues a DFS2082 message to the client to indicate that the transaction terminated with no reply.

This enhancement enables you to convert send-then-commit (CM1) transactions into CM0 transactions without having to modify your applications.

Send/receive programming model

Use the send/receive programming model to run an IMS response mode transaction.

To run a transaction in IMS, a Java application runs a SYNC_SEND_RECEIVE interaction. In your application, provide the following values for the IMSInteractionSpec object that is used by the execute method of the Interaction object:

- ▶ A value of SYNC_SEND_RECEIVE for the interactionVerb property
- ▶ A value of 0 or 1 for the commitMode property

However, the SYNC_SEND_RECEIVE interaction processing is different for shareable and dedicated persistent socket connections. Depending on the type of socket connections, the processing model is different in either a normal processing of the transaction, or when an error or an execution timeout occurs.

With IMS TMRA V12 and later, if you convert an IMS V12 send-then-commit (CM1) application that expects a response to a commit-then-send (CM0) application that does not receive a response, set the IMSInteractionSpec CM0Response property to true. When this property is set for a CM0 transaction, if the IMS application does not reply to the IOPCB or complete a message switch to another transaction, IMS OTMA issues a DFS2082 message to the client, regardless of the transaction response mode.

Shareable persistent socket processing model

Shareable persistent socket connections are connections that can be used for commit mode 1 and commit mode 0 interactions. The following scenarios describe the SYNC_SEND_RECEIVE interaction on a shareable persistent socket during normal processing, error processing, and execution timeout.

Normal processing scenario

The IMS TMRA, with the application server, obtains either an available connection from the connection pool or creates a connection. The IMS TMRA, as part of initializing a new connection, generates a client ID for the connection. The generated client ID identifies the socket connection, and for commit mode 0 interactions, the tpipe and associated OTMA asynchronous hold queue.

The IMS TMRA ensures that a socket is associated with the connection and sends the request with input data to IMS Connect by using that socket. IMS Connect then sends the message to IMS, where IMS runs the transaction and returns the output message.

For commit mode 0 interactions, when it receives the output message, the IMS TMRA sends an ACK message to IMS, which signals IMS to discard the output from the IMS queue. When the client application closes the connection or terminates, the connection is returned to the connection pool for reuse by other commit mode 0 or commit mode 1 interactions.

Error processing scenario

All errors result in a resource exception being thrown to the client application. In addition, some errors result in the socket being disconnected by IMS Connect. For commit mode 0 interactions, an exception means that the output message cannot be delivered to the client application. However, following the exceptions, undelivered output messages for commit mode 0 interactions on shareable persistent socket connections can be retrieved if the SYNC_SEND_RECEIVE interaction specifies that undelivered output is to be rerouted to a specific destination. To have an undelivered output message rerouted to a specific destination, the following additional properties must be specified in the IMSInteractionSpec object that is passed on the SYNC_SEND_RECEIVE interaction:

- ▶ Set the purgeAsyncOutput property to false so that undelivered output is not purged.
- ▶ Set the reRoute property to true, and specify a reroute destination in the RouteName property.

To retrieve undelivered output from a reroute destination, a separate client application issues a SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_NOWAIT or SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT interaction on a dedicated persistent socket connection. The client application provides the reroute destination as the client ID of the interaction.

Alternatively, to retrieve undelivered output from a reroute destination, a separate client application issues a SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_NOWAIT, or SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT interaction on a shareable persistent socket connection by specifying the alternative client ID. Using the alternative client ID, a client application can retrieve undelivered asynchronous output messages from any tpipe.

The default value of the purgeAsyncOutput property is true. When the value of the purgeAsyncOutput property is true, the following output messages are purged:

- ▶ Undelivered output message inserted to the I/O Program Communications Block (I/O PCB) by the primary IMS application program
- ▶ Output messages inserted to the I/O PCB by secondary IMS application programs invoked by program-to-program switches

When the purgeAsyncOutput property is set to false, the reroute destination must be specified.

Execution timeout scenario

If an execution timeout occurs, the socket connection remains open but the output message is not delivered to the client application. However, following an execution timeout exception, undelivered output messages for commit mode 0 interactions on shareable persistent socket connections can be retrieved in either of the following two ways:

- ▶ The same client application that issued the SYNC_SEND_RECEIVE interaction can issue a SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_NOWAIT or
- ▶ The same client application that issued the SYNC_SEND_RECEIVE interaction can issue a SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT interaction.

The undelivered output message can be rerouted to a specific destination, as described in “Error processing scenario” on page 231.

When the client application closes the connection or terminates, the connection is returned to the connection pool so it can be reused by other commit mode 0 or commit mode 1 interactions.

Support for IMS V12 RACF return codes

In IMS V12, if a RACF security failure occurs, IMS Connect includes a 2-byte return code from the RACF **RACROUTE REQUEST=VERIFY** command. This version of the IMS TMRA is enhanced to take advantage of the enhancement so the RACF security failure reason code is available to IMS TMRA to assist troubleshooting.

6.3 Installing the IMS TM Resource Adapter

The latest version of the IMS TMRA can be found at the following website:

<http://www.ibm.com/software/data/ims/ims/components/tm-resource-adapter.html>

Consideration: Synchronous callout support requires IMS TMRA V11 or V10.3.

For more information about the IMS TMRA, see *IMS TM Resource Adapter User's Guide and Reference*, SC19-1211-02.

At the website, download the IMS TMRA .rar file and use the WebSphere Application Server administrative console to deploy it. To access the administrative console, perform the actions that are shown in Figure 6-7 on page 233, which are outlined here:

1. Start WebSphere Application Server.
2. In the Servers view, right-click the server and select **Administration** → **Run Administrative Console**. The Administrative Client logon window opens.
3. If the server is secured, specify the user ID and password to access the administration console.
4. Click the **Login** button.

Note: With Rational Developer for z, WebSphere Application Server is not installed. You must install the WebSphere Application Server that comes with the Rational Application Developer product before you install Rational Developer for z.

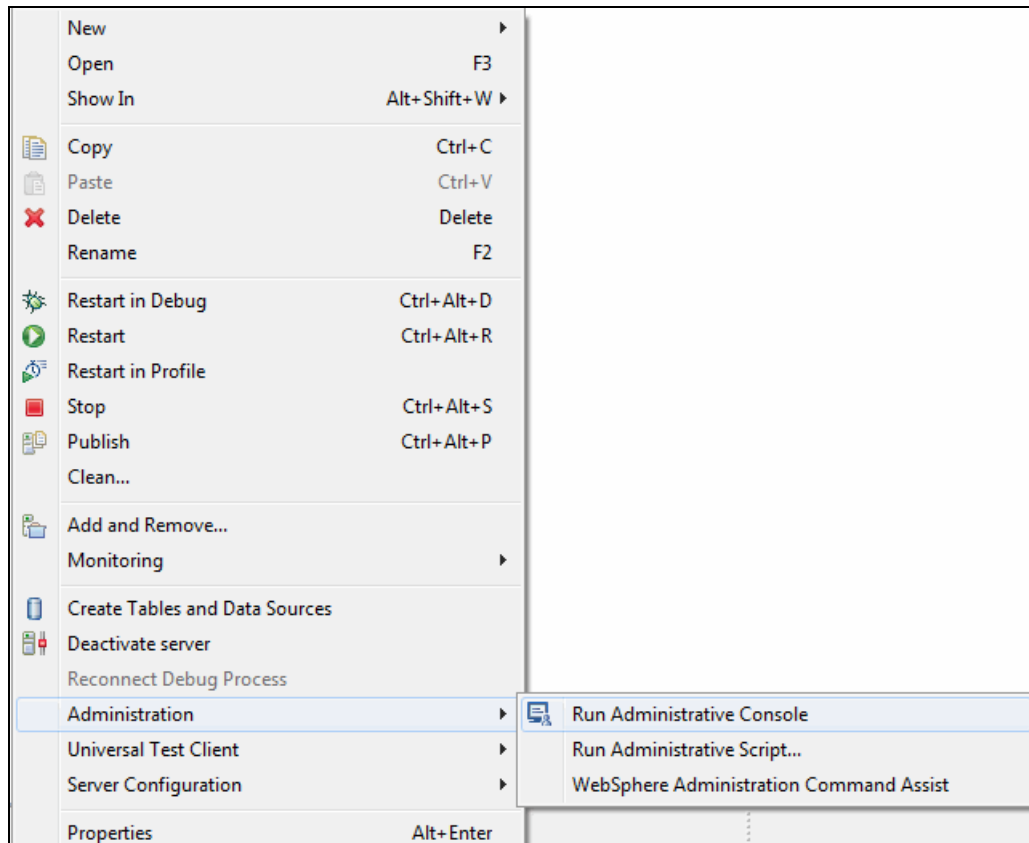


Figure 6-7 Run Administrative Console

Here are some important steps to consider when you install the IMS TMRA:

1. If you are installing the IMS TMRA archive (.rar file) on a generic application server, more class libraries might be needed for use with the IMS TMRA. For class libraries that are not included in the IMS TMRA installation archive, contact your product representative for the development environment of your choice for information regarding the licensing and location of any needed JAR files.
2. When you use the Install RAR dialog to install a .rar file, the scope you define on the Resource Adapters page has no effect on where the .RAR file is installed. You can install .RAR files only at the node level, which you specify on the Install RAR page. To set the scope of a .rar file to a specific cluster, or server, after you install the .rar file at each node level, create a copy of the .rar file with the appropriate cluster or server scope.

You must have installed the IMS TMRA .rar file in a file system that is accessible to your WebSphere Application Server.

Start the WebSphere Application Server and log in to the administrative console to deploy the .rar file on a WebSphere Application Server:

1. In the navigation pane in the administrative console (also known as the Integrated Solutions Console), click **Resources** → **Resource Adapters** → **Resource adapters**, as shown in Figure 6-8.

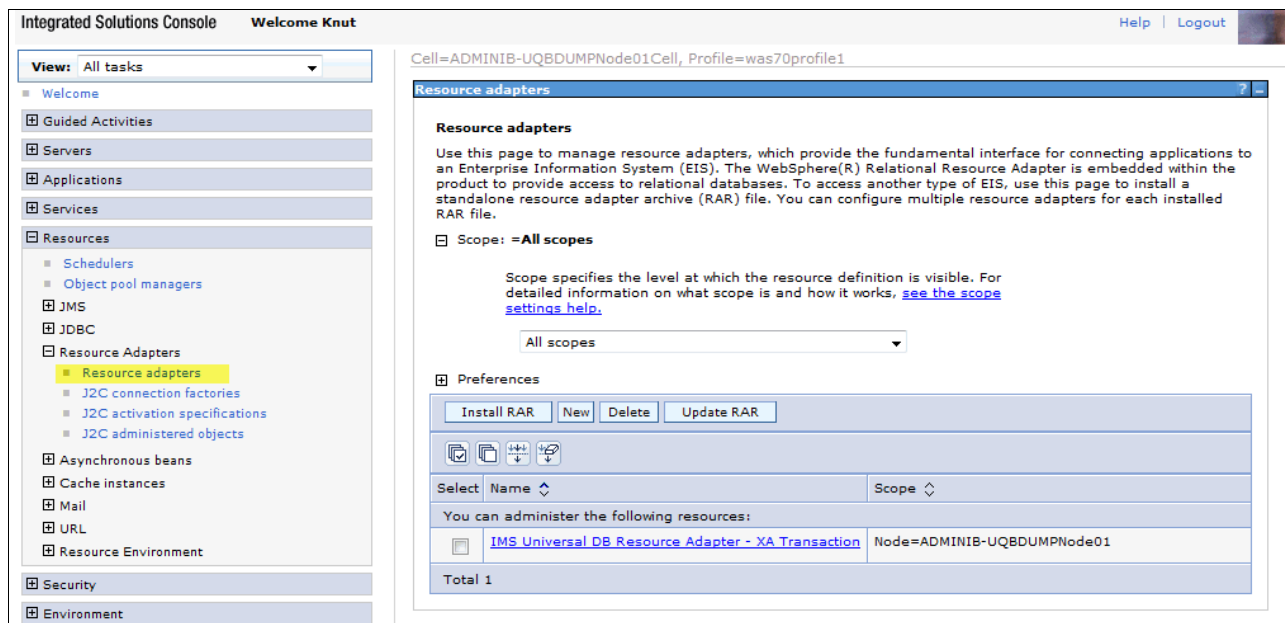


Figure 6-8 Resource adapter configuration in WebSphere Application Server

2. If a .rar file that you no longer want to use is installed, delete it first and then stop and restart the WebSphere Application Server. Repeat step 1 to return to the resource adapter configuration page.
3. In the topic pane, click **Install RAR**. The Install RAR File window opens.
4. Click **Local file system** or **Remote file system**, depending on the location of the .rar file.
If you choose **Local file system**, click **Browser** to locate and specify the IMS TMRA .rar file that is installed, for example,
install_path/IBM/IMS/IC0xx/Vxxxx/JCA15/imsxxxx.rar.
5. Click **Next**.
6. On the Configuration page, enter a name for the .rar file, for example, *imstmravxxxx*. Click **OK**.
7. Optionally, create a copy of the .rar file with a different scope level. After you install the .rar file at each node level, you can create another copy of the file that has a specific server or cluster as the scope for that file. For more information about this step, see the “Installing a resource adapter archive” topic in the WebSphere Application Server information center, found at:
<http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp>
8. To save your results in the master configuration, click **Save**.

6.4 Callin request support

Support for external clients to request services from IMS was available for many years. This function can be requested from all Java EE artifacts, Servlets, EJBs, Business Processes, and Mediation Modules. The non-*conversational* transaction type is the most simple and most frequently used service. Conversational transactional flow is also supported, but does not always work well in SOA scenarios.

When a Java application submits a transaction request to IMS through IMS TMRA, IMS Connect sends the transaction request to OTMA by using cross-system coupling facility (XCF) communications, and the transaction runs in an IMS dependent region. The response is returned to the Java application using the same path.

If access is engaged through a deployed IMS TMRA in the WebSphere Application Server Java EE container, the Common Client Interface (CCI) contract is honored. Figure 6-9 shows the container component contract with respect to the other components within the container.

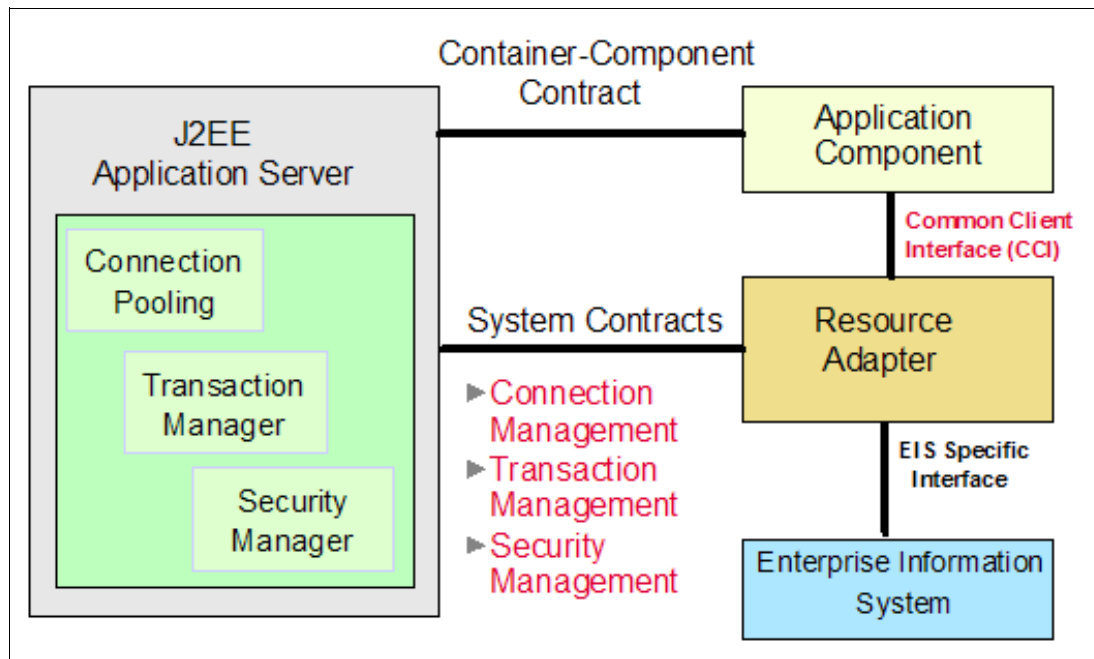


Figure 6-9 Common Client Interface contract

The CCI contract is responsible for three processing components:

- ▶ Connection management (including connection pooling). This implies that to establish the connection with IMS, a “Managed Connection Factory” is used.
- ▶ Security management through the passing of user IDs.
- ▶ Transaction management through the control of processing units of work.

The interaction with IMS Connect and IMS is based on four Java objects:

- Connection

This object represents the connection. It is obtained from the *ConnectionFactory* that is in a WebSphere Application Server. Figure 6-10 presents the properties that can be defined in the Connection Factory

2C connection factories

[J2C connection factories](#) > [CFIMSLN](#) > Custom properties

Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.

Preferences

Name	Value	Description	Required
SSLKeyStoreName		Name (full path) of SSL keystore for client certificates/private keys	false
SSLKeyStorePassword		Password of SSL keystore for client certificates/private keys	false
UserName		Default name of the user to be authorized	false
HostName	wtsc67.itso.ibm.com	TCP/IP host name of the target IMS Connect	false
SSLTrustStorePassword		Password of SSL keystore for trusted certificates	false
SSLTrustStoreName		Name (full path) of SSL keystore for trusted certificates	false
IMSConnectName		Name of the target IMS Connect - for Local Option only	false
Password		Default password of the user	false
TraceLevel	1	Level of information to be traced	false
DataStoreName	IMSL	Name of the target IMS datastore	false
SSLEncryptionType	Weak	The type of cipher suite to be used for encryption	false
SSLEnabled	FALSE	Indicates if SSL is enabled for this connection factory	false
GroupName		Default name of the IMS group of the user	false
PortNumber	9999	Target TCP/IP port number of IMS Connect	false
CMODedicated	FALSE	Indicates if sockets are dedicated to specific CM0 clients	false
Total 15			

Figure 6-10 Custom properties of IMS Connection Factory

Define the factory with the WebSphere Application Server Administration Console, and label it with a JNDIName for “look-up”.

Note: The factory can also be instantiated at run time if no container (WebSphere Application Server) is available, which is also true for Java Batch and stand-alone programs. In that case, the factory must be instantiated at run time.

- ConnectionSpec

This object provides some additional connection properties for the connection, which are related to security and identification. For more information, see Figure 6-11 on page 237.

Property Name - Type	Variable Name
Connection Spec	
password - String	myPassword
groupName - String	myGroupName
userName - String	myUserName
clientID - String	myClientID
Interaction Spec	
purgeAsyncOutput - boolean	myPurgeAsyncOutput
imsRequestType - int	myImsRequestType
socketTimeout - int	mySocketTimeout
mapName - String	myMapName
asyncOutputAvailable - boolean	myAsyncOutputAvailable
syncLevel - int	mySyncLevel
useConvID - boolean	myUseConvID
ltermName - String	myLtermName
interactionVerb - int	myInteractionVerb
convEnded - boolean	myConvEnded
commitMode - int	myCommitMode
convID - String	myConvID
altClientID - String	myAltClientID
reRoute - boolean	myReRoute
reRouteName - String	myReRouteName
executionTimeout - int	myExecutionTimeout

Figure 6-11 ConnectionSpec and InteractionSpec

► Interaction

This object is the coordinator of the interaction that is run with IMS. Its run method triggers the interaction.

► InteractionSpec

This object specifies the details of the interaction. For more information, see Figure 6-11.

The InteractionVerb is probably the most important parameter, which indicates what action IMS Connect must take. The TM Resource client is always the contacting partner, even if IMS must send output.

Here are the available verbs:

► SYNC_SEND (value 0)

Sends a request to IMS when a response is not expected, that is, it performs a send only interaction.

► SYNC_SEND_RECEIVE (value 1)

Used for the single iteration of a non-conversational IMS transaction and for each iteration of a conversational IMS transaction.

Note: SYNC_RECEIVE (value 2) is not supported by IMS Connector for Java.

► SYNC_END_CONVERSATION (value 3)

Forces the end of an IMS conversational transaction.

- ▶ **SYNC_RECEIVE_ASYNCOUTPUT** (value 4)
Retrieves asynchronous output messages. With this type of interaction, the Java client can receive only a single message. If there are no messages in the IMS OTMA Asynchronous Queue for the client ID when the request is made, no further attempts are made to retrieve the message. No message is returned and a timeout occurs after the length of time that is specified in the `executionTimeout` property of the **SYNC_RECEIVE_ASYNCOUTPUT** interaction.
- ▶ **SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_NOWAIT** (value 5)
Retrieves asynchronous output messages. With this type of interaction, the Java client can receive only a single message. If there are no messages in the IMS OTMA Asynchronous Queue for the client ID when the request is made, no further attempts are made to retrieve the message. No message is returned and a timeout occurs after the length of time that is specified in the `executionTimeout` property of the **SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_NOWAIT** interaction.
- ▶ **SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT** (value 6)
Retrieves asynchronous output messages. With this type of interaction, the Java client can receive only a single message. If there are no messages in the IMS OTMA Asynchronous Queue for the client ID when the request is made, IMS Connect waits for OTMA to return a message. IMS Connect waits the length of time that specified in the `executionTimeout` property of the **SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT** interaction before returning an exception.

The Generated Client ID function: In IMS V11, the IMS TMRA is enhanced to take advantage of a new function in IMS Connect, the Generated Client ID function, which ensures the uniqueness of each socket.

This enhancement eliminates the requirement for IMS TMRA to use different IMS Connect ports for instances of distributed WebSphere Application Server. IMS Connect can use the Generated Client ID function to ensure the uniqueness of each socket and allow all instances of WebSphere Application Server to specify the same IMS Connect TCP/IP port.

6.5 Callout request support

Although we call this pattern *callout*, it starts with a *callin* invitation (**SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT**) from the IMS TMRA partner, which is a Java EE application.

Asynchronous callout was made available with IMS V10. An SPE on IMS V10 (APAR PK74168) added the synchronous capability, which requires WebSphere Application Server V6.1 as the minimum level.

At least two data flows exist in a callout pattern with the WebSphere Application Server. A third inbound data flow to IMS TMRA is required if IMS expects a response.

1. **SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT**
2. Receive callout request
3. **SYNC_SEND** (send response) (optional)

Figure 6-12 illustrates this callout data flow.

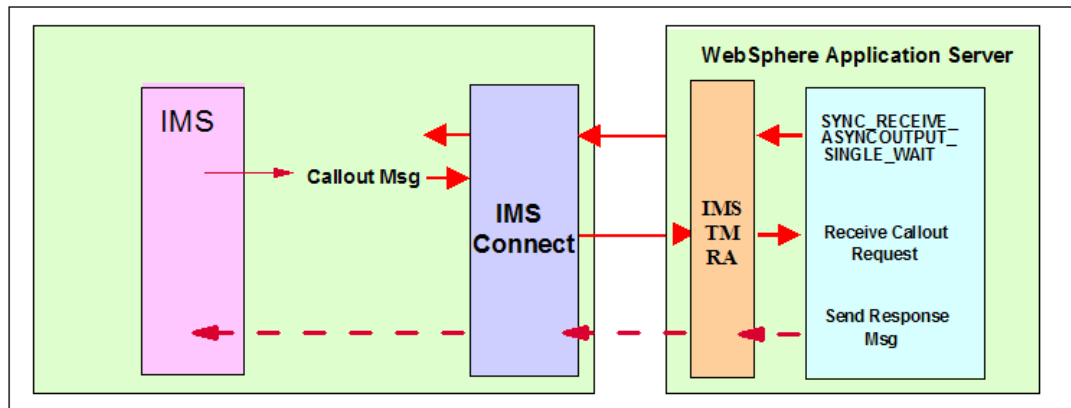


Figure 6-12 Callout data flow

OTMA destination routing descriptors are used by customers who implement this callout function or who implement message switches from OTMA to non-OTMA destinations and do not want to code the OTMA routing exits (DFSYPX0 and DFSYDRU0). If an OTMA descriptor is used with the callout function, Destination Routing Descriptors in the DFSYDTx member of IMS.PROCLIB must be created that specify the appropriate properties for the callout destination. The properties depend on the type of callout.

There are two types of callout requests:

- ▶ Asynchronous: The request is sent, but the sending IMS program does not wait for the response, which is returned as a separate IMS transaction invocation if necessary. This happens through an ISRT call to an ALTPCB destination.
- ▶ Synchronous: The sending IMS program waits for the response from the send / receive call flow. Also, a timeout value must be provided. The request is performed through the new "ICAL" call function to an alternative destination, which is described through an OTMA descriptor.

A new optional keyword **SYNTIMER** is introduced to the OTMA descriptor so that the user can specify a time value to wait in hundredths of seconds, which can be 1 - 900000, for synchronous call process to complete. Example 6-2 shows an example of the **SYNTIMER** keyword.

Example 6-2 OTMA descriptor keyword SYNTIMER

```
D WASDEST1 TYPE=IMSCON TMEMBER=HWS1 TPIPE=WASTP1 SYNTIMER=5000
```

6.5.1 Destinations for callouts through IMS TM Resource Adapter

Two types of Enterprise beans (EJBs) can be solicited for serving the callout request. They take the initiative to pull the callout message from a hold queue with a SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT interaction.

Here are the types of EJBs that you can use:

► Stateless Session bean (SLSB)

Stateless Session beans are distributed objects, server-side Java EE components that do not have a state that is associated with them, thus allowing concurrent access to the bean. Stateless session EJBs display the following behavior:

- Provide a relatively short-lived single use service.
- Do not maintain a state on behalf of the client and do not survive EJB server crashes.
- Any two instances of the same stateless session EJB type are always identical, and each instance can be shared by multiple clients.

For more information about this subject, see the WebSphere information center at the following website:

http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp?topic=/com.ibm.websphere.base.doc/info/welcome_base.html

► Message Driven bean (MDB)

Message-driven beans (MDBs) are stateless, server-side Java EE components for processing asynchronous messages. An MDB bean is an asynchronous bean that is activated by message deliveries and can consume and process messages concurrently.

An MDB can be used as a listener to receive callout requests that are retrieved by IMS TMRA. Then, the MDB can route the message to the appropriate EJB or other application within the WebSphere Application Server for further processing.

6.5.2 Asynchronous callout requests

IMS application programs that are running in IMS-dependent regions can send outbound messages to request services from a WebSphere Application Server Enterprise bean. The request for services or data is referred to as a *callout request*. When an IMS application program makes a callout request, IMS can be viewed as a client in a client/server relationship where the server is the external application to which IMS is making the callout request. If the request is treated as an asynchronous request, there is no answer (send only), or a response is sent back as a new transaction.

An IMS configuration that supports callout includes these functions and components:

- The IMS application program that issues the callout request and, if necessary, the IMS application program that processes the callout response.
- Optionally, an IMS database to store the data necessary to correlate the callout response to the initiating IMS client.
- An OTMA instance that contains the following items:
 - Asynchronous hold queue
 - OTMA ALTPCB destination descriptor
 - OTMA routing exits
 - OTMA asynchronous hold queue security
 - Optionally, an OTMA Resume TPIPE Security exit routine (DFSYRTUX)
- IMS Connect instance.

- ▶ WebSphere Application Server:
 - IMS TMRA running on WebSphere Application Server
 - WebSphere Application Server EJBs
 - The external application program
- ▶ Optionally, the use of RACF

Each Java application using IMS TMRA that processes a callout request externally can be configured to “listen” to the OTMA asynchronous hold queue by issuing a looping **RESUME TPIPE** call with an appropriate wait time. The Java application program might also preserve the correlation data and send it back with the callout response.

OTMA routing of callout requests is required to route the ISRT alternate_pcb call to the appropriate tpipe. The OTMA routing is performed by the OTMA ALTPCB destination descriptor, or the OTMA Destination Resolution exit routine (DFSYPX0) and the OTMA User Data Formatting exit routine (DFSYDRU0).

In WebSphere Application Server, the listening Java (EJB) code can be a Stateless Session bean (SLSB) or the listener of a Message Driven bean (MDB).

Securing the retrieval of asynchronous output and callout messages

The asynchronous output and the asynchronous callout programming models of the IMS TMRA allow you to retrieve asynchronous output or callout request messages from the hold queue. To ensure that only an authorized user can retrieve an asynchronous output message or a callout request from the hold queue, you can specify the user ID together with the tpipe name that is contained in the **SYNC_RECEIVE_ASYNCOUTPUT_*** command message. Authorization is performed by IMS OTMA when the message is retrieved from the hold queue.

The user ID and password information can be specified in the IMSConnectionSpec object, in the authentication alias that the IMS ConnectionFactory uses, or the application’s deployment descriptor. For more information about OTMA security, see the ‘Specifying OTMA security’ and ‘Securing messages on the asynchronous hold queue’ sections in *IMS Version 10 Communications and Connections Guide*, SC18-9703.

Asynchronous callout to an SLSB

Figure 6-13 shows the interaction with a Stateless Session EJB. Because the activity that is associated with the setup within the SLSB and IMS application are not directly tied together, begin the flow within the SLSB running in WebSphere Application Server. This is the opposite of the examples in 5.6, “Synchronous callout with IMS SOAP Gateway” on page 207. If a callout request did not arrive at the waiting SLSB, it waits for the next available callout message or until a timeout occurs.

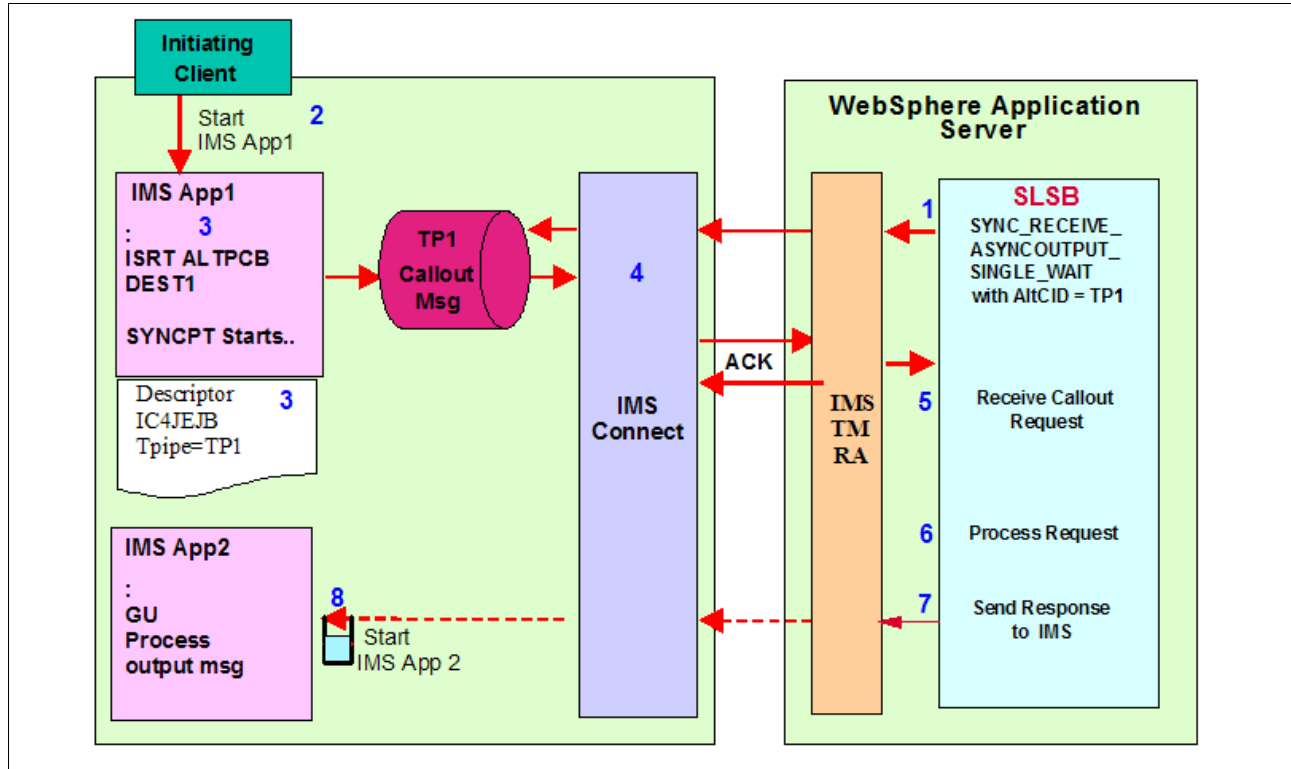


Figure 6-13 Message flow of asynchronous callout to SLSB using IMS TMRA

Here is the message flow of an asynchronous callout to SLSB using the IMS TMRA:

1. In preparation for the flow, the EJB application in a WebSphere Application Server starts and obtains a shareable persistent connection to IMS Connect through the IMS TMRA. It issues a SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT interaction, specifying the tpipe name as the value for the alternative client ID, and sets a large timeout value. IMS TMRA then issues a **RESUME TPIPE** request to the tpipe and waits for the callout request from IMS Connect.
2. An initiating client, such as a terminal or an IMS Connect or OTMA client, starts an IMS application
3. The IMS application issues an ISRT ALTPCB call to an OTMA destination routing descriptor, which contains the destination tpipe name. The callout request message is queued on this tpipe.
4. When the callout request is available in the tpipe, IMS Connect delivers the callout message to IMS TMRA.
5. IMS TMRA receives the callout request message and presents the callout request to the EJB.
6. The EJB processes the callout request.

7. If the EJB has response data to send back to IMS, the EJB issues a normal IMS send_only transaction request through IMS TMRA and IMS Connect to IMS.
8. A second IMS application is scheduled to handle the asynchronous reply.

To prepare a Java application for inbound requests from an IMS application program, you must specify the appropriate interaction verb and the asynchronous hold queue name in the Java application, and specify a timeout value.

Specify the asynchronous hold queue name:

```
InteractionSpec.setAltClientID(calloutQueueName);
```

Set the interactionVerb property to SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT:

```
interactionSpec.setInteractionVerb(com.ibm.connector2.ims.ico.  
IMSInteractionSpec.SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT);
```

Specify an execution timeout value (in milliseconds) that you want to wait for a callout request message in the hold queue:

```
interactionSpec.setExecutionTimeout(3600000);
```

In this example, we specify a large execution timeout value. A large execution timeout value helps minimize the looping that is required in a callout EJB when there might be long periods when no callout requests occur.

Sample code that is related to the asynchronous callout to an SLSB can be found in A.1, “ICAL Synchronous Program Switch COBOL program” on page 430.

Asynchronous callout to an MDB

Message Driven beans (MDB) allow your application to asynchronously receive messages that are delivered to a JMS destination. Figure 6-14 shows the interaction with a Message Driven bean.

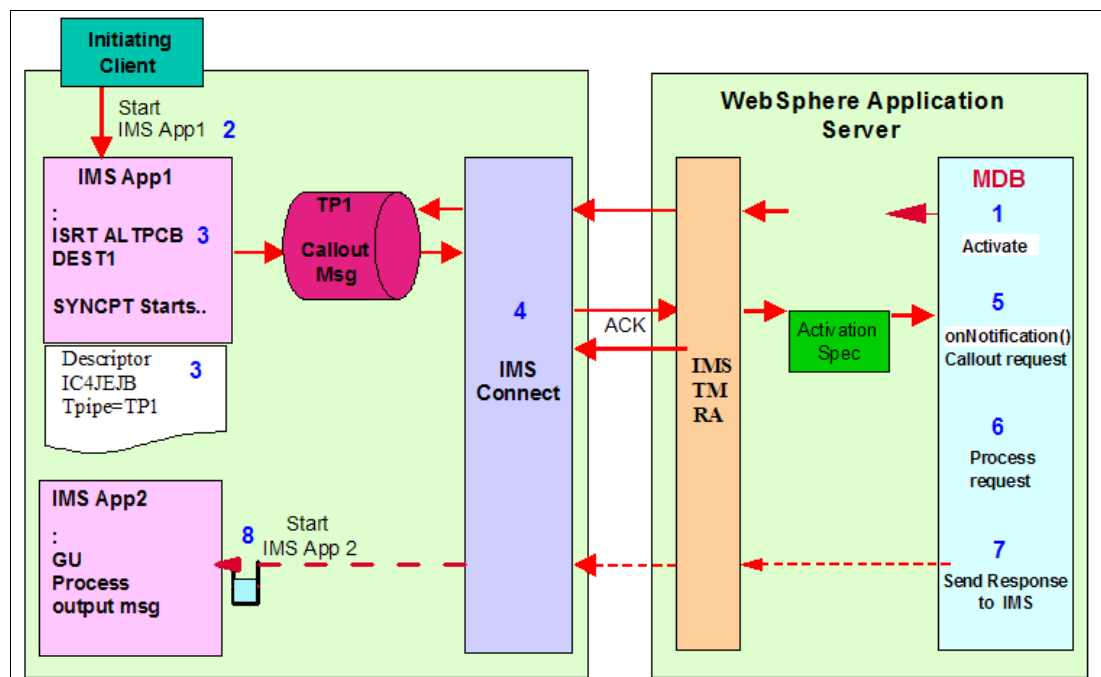


Figure 6-14 Message flow of asynchronous callout to MDB using IMS TMRA

Here is the message flow of an asynchronous callout to MDB using IMS TMRA:

1. When the Java EE application containing the deployed MDB is started in the WebSphere Application Server through the ActivationSpec and the underlaying listening software, a shareable persistent connection to IMS Connect through IMS TMRA is obtained. The Java EE application issues a SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT interaction, which specifies the tpipe name as the value for the alternative client ID and sets a large timeout value. IMS TMRA, in turn, issues a **RESUME TPIPE** request to the tpipe and waits for the callout request from IMS Connect. The addressing information is pulled from properties in the EJB (MDB) deployment descriptor.
2. An initiating client, such as a terminal, IMS Connect, or OTMA client, starts the scheduling of an IMS application.
3. The IMS application issues an ISRT ALTPCB call to an OTMA destination routing descriptor, which contains the destination tpipe name. The callout request message is queued in this tpipe.
4. When the callout request is available in the tpipe, IMS Connect delivers the callout message to IMS TMRA.
5. IMS TMRA receives the callout request message and passes the request to the MDB in the **onNotification()** method.
6. The MDB processes the callout request.
7. If response data must be sent back to IMS, the MDB issues a normal IMS transaction with a SYNC_SEND request to the appropriate IMS application with the transcode and response data.
8. In this case, IMS App2 is initiated by the receipt of the response from the MDB.

Sample MDB code is shown in A.3, “Asynchronous callout to a Message Driven bean” on page 437.

6.5.3 Synchronous callout requests

Figure 6-15 on page 245 shows that with the IMS callout support, IMS applications can be a client and server. IMS provides bidirectional access between IMS applications and external application and servers. The IMS application program can perform the following functions:

- ▶ Call out to a user-written IMS Connect client.
- ▶ Call out to WebSphere EJB/MDB using IMS TMRA.
- ▶ Call out to a web service provider using IMS SOAP Gateway.

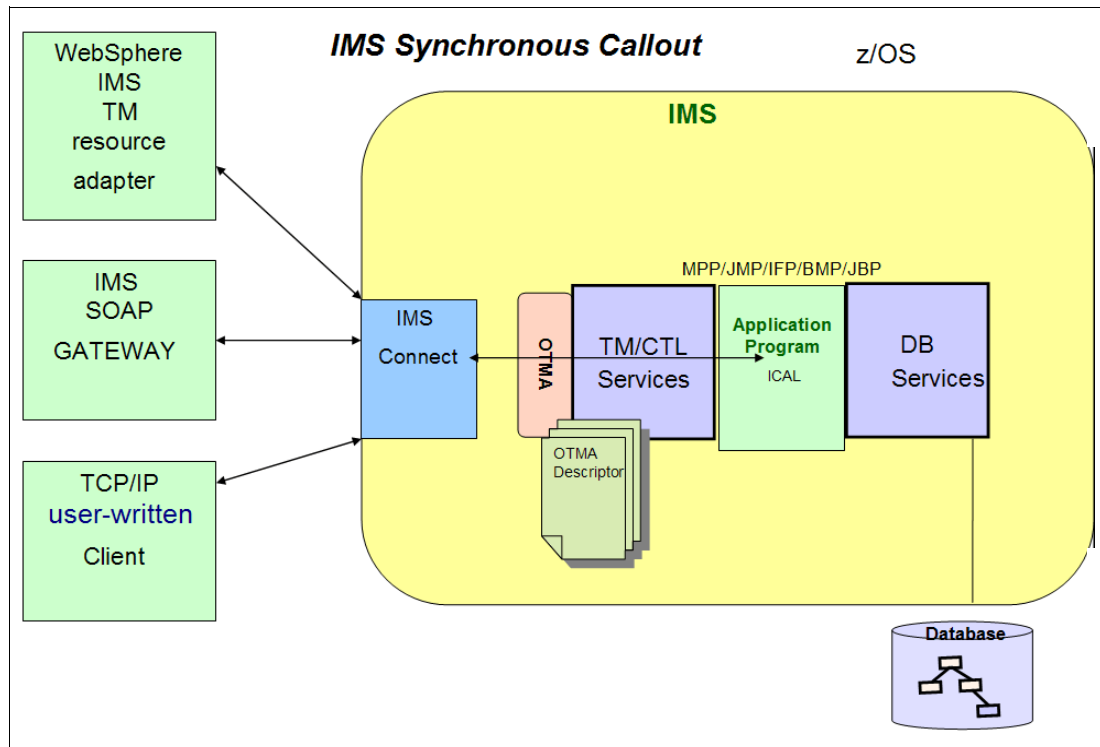


Figure 6-15 Synchronous callout interfaces

A new DLI call, the ICAL SENDRECV synchronous call function, allows an IMS application program to communicate with other external applications synchronously. An IMS Call (*ICAL*) sends a synchronous request for data or services from an IMS application program running in the IMS TM environment to a non-IMS application program or service running in a distributed environment. ICAL processing does not use the IMS message queue.

Here is the format of this new DLI call:

```
CALL 'AIBTDLI' USING function, aib,i/o area
```

Here are the parameters:

- function: Set to "ICAL"
- aib

Specifies the application interface block (AIB) to be used for this call.

- AIBID

Eyecatcher. This 8-byte field must contain DFSAIBbb.

- AIBLEN

AIB length. This field must contain the actual length of the AIB that the application program obtained.

- AIBSFUNC

This sub function code field must contain one of the listed 8-byte sub function codes: SENDRECV. The IMS application program uses this sub function for synchronous program to program communication.

- AIBRSNM1

This resource name field contains the OTMA descriptor name. AIBRSNM1 is an 8-byte alphanumeric left-align field padded with blanks.

- AIBRSFLD

This 4-byte field is used by the IMS application program to specify a time value to wait in one-hundredths of a second for the synchronous call process to complete. This value overrides the value that is specified in the OTMA descriptor. The minimum value is 0 and the maximum value is 999999. The system default is 10 seconds.

- AIBOALEN

This request area length is an input and an output parameter. As an input parameter, this 4-byte field must contain the length of the input request area that is specified in the call list. As an output parameter, this field is updated only when partial data is returned (AIB return code x'100', AIB reason code x'00C'). In the case of partial data that is returned, this field contains the length of the response message. For any return code other than partial data, this field is unchanged.

- ▶ i/o area

Specifies the name of the I/O area that is used for the call. The I/O area must be large enough to contain all the returned data.

The ICAL synchronous call function provides the ability for an IMS application program to communicate with other application programs in a synchronous manner using the new OTMA message processing function. Because the IMS message queue is not used, the 32 K message segment restriction was removed.

The ICAL synchronous call function is supported only by using the AIBTDLI interface in IMS TM runtime environments for IFP, MPP, BMP, JMP, and JBP regions.

Attention: Coordinated two-phase commit between the IMS application program and the external application program is not supported in the IMS Version 10 SPE. Also, a Shared Queues back-end IMS that is not connected to IMS Connect cannot issue this ICAL.

Here are the functions that OTMA provides as part of its support for the IMS synchronous callout Version 10 SPE:

- ▶ Flows the synchronous callout request and receives the response.
- ▶ Enhances the OTMA output descriptor.
- ▶ Provides the timeout function for the ICAL call.
- ▶ Provides the OTMA protocol updates so that the synchronous callout request and the response can be recognized by both IMS Connect and OTMA.
- ▶ Provides a simplified OTMA TPIPE log record to track the input and output flow of a synchronous callout message.
- ▶ Provides the OTMA command updates.

OTMA connects the ICAL call to the IMS Connect client applications so that a synchronous callout request can be delivered to them and the response can be given back to the IMS application. This synchronous callout request is delivered through the OTMA Resume TPIPE method. If the request is larger than 32 KB, it is delivered to IMS Connect using a multi-segment flow. The size of each message segment is equal or less than 32 KB. IMS Connect assembles the segments and delivers them as one complete message to IMS Connect client applications, such as IMS TMRA.

Synchronous callout to an SLSB

Figure 6-16 shows the interaction with a Stateless Session EJB.

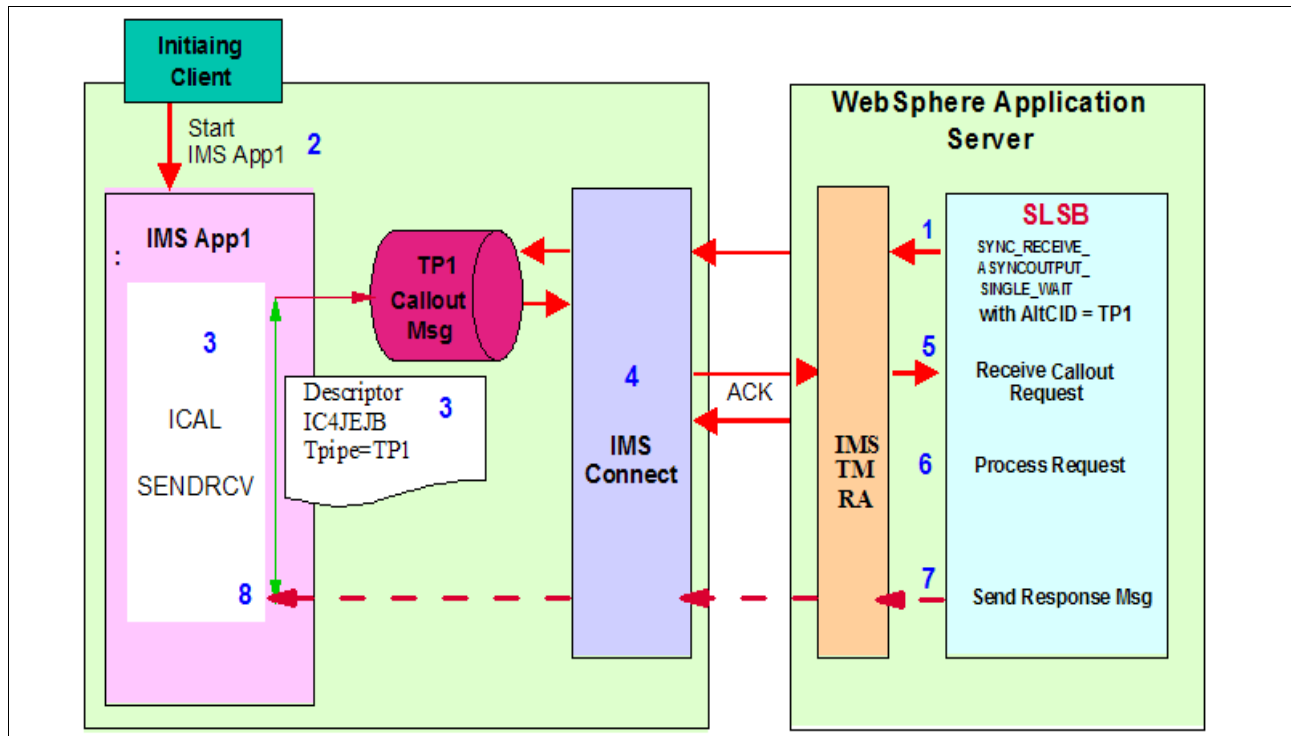


Figure 6-16 Message flow of a synchronous callout to SLSB using IMS TMRA

Here is the message flow of a synchronous callout to SLSB using IMS TMRA:

1. The SLB application in WebSphere Application Server starts and obtains a shareable persistent connection to IMS Connect through IMS TMRA. It issues a SYNC_RECEIVE_ASYNCOUPTUT_SINGLE_WAIT interaction, specifying the tpipe name as the value for the alternative client ID, and sets a large timeout value. IMS TMRA in turn issues a **RESUME TPIPE** request to the tpipe and waits for the callout request from IMS Connect.
2. An initiating client, such as a terminal, an IMS Connect, or an OTMA client, starts an IMS application.
3. The IMS application issues an ICAL call to an OTMA destination routing descriptor, which contains the destination tpipe name. The callout request message is queued in this tpipe.
4. When the callout request is available in the Tpipe, IMS Connect delivers the callout message to the IMS TMRA.
5. IMS TMRA receives the callout request message and returns the callout request to the SLSB.
6. The EJB processes the callout request.
7. A response message for a synchronous callout request returns to OTMA as a special form of SYNC_SEND message. This type of Send-Only message has no transcode and can be a regular response data or error information to be passed to the ICAL call. The response message can also be a multi-segment message. OTMA assembles the multi-segment messages into a single message to be returned to the IMS application.

The correlation between the request from IMS and response to IMS is assumed by the correlation token in interactionSpec of the SYNC_RECEIVE_ASYNCOUPTUT_SINGLE_WAIT and SYNC_SEND interaction.

8. The waiting IMS App1 is presented the response.

Sample code for the SLSB is in A.4, “Synchronous callout to a Stateless Session bean” on page 440.

Synchronous callout to an MDB

MDBs allow your application to synchronously receive messages that are delivered to a JMS destination. Figure 6-17 shows the interaction with an MDB.

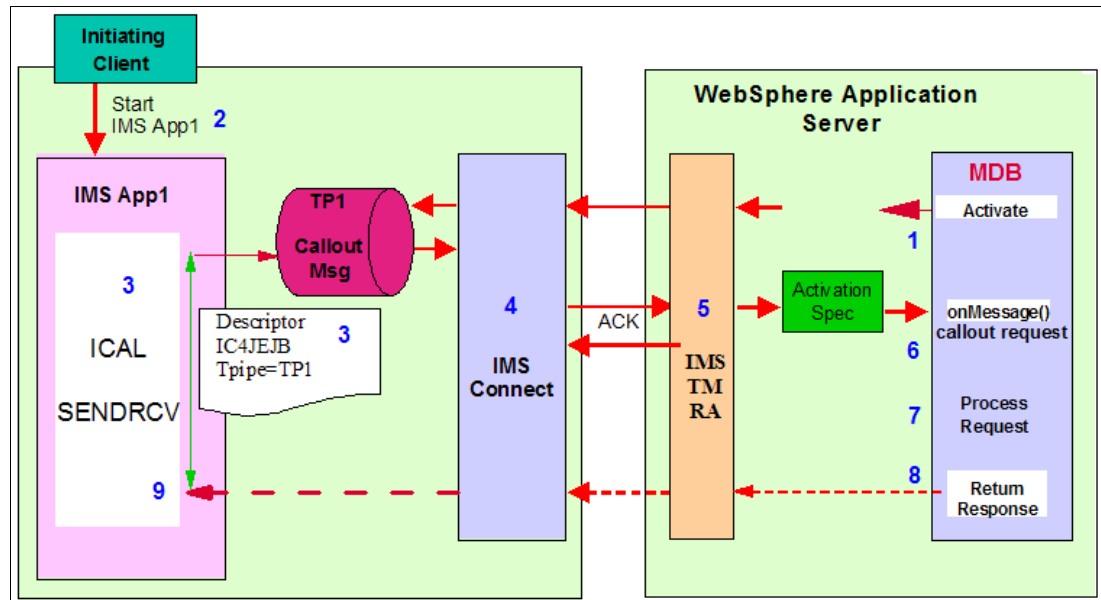


Figure 6-17 Message flow of a synchronous callout to MDB using IMS TMRA

Here is the message flow of a synchronous callout to MDB using IMS TMRA:

1. When the Java EE application that contains the deployed MDB is started in WebSphere Application Server through the ActivationSpec and the underlying listening software, a shareable persistent connection to IMS Connect is obtained through IMS TMRA. It issues a SYNC_RECEIVE_ASYNCOUPTUT_SINGLE_WAIT interaction, specifying the tpipe name as the value for the alternative client ID, and sets a large timeout value. IMS TMRA issues a **RESUME TPIPE** request to the tpipe and waits for the callout request from IMS Connect. The addressing information is pulled from the properties in the MDB deployment descriptor.
2. An initiating client, such as a terminal, IMS Connect, or an OTMA client starts the scheduling of an IMS application.
3. The IMS application issues an ICAL call to an OTMA destination routing descriptor, which contains the destination tpipe name.
4. The callout request message is queued in this tpipe.
5. When the callout request is available in the tpipe, IMS Connect delivers the callout message to IMS TMRA.
6. IMS TMRA receives the callout request message and passes the callout request to the MDB in the **onMessage()** method.

7. The MDB processes the callout request.
8. Response data is returned as an object of type `Javax.resource.cci.Record` with the output data.
9. The waiting IMS App1 is presented the response.

Sample code for the MDB is in A.5, “Synchronous callout to a Message Driven bean” on page 442.

Orphaned IMS conversations

When a conversation is not ended explicitly, it continues to exist in the system as an orphaned conversation, and the associated IMS storage continues to be allocated to that conversation.

An IMS conversation is ended explicitly in one of two ways:

- ▶ The IMS application inserts blanks in to the SPA before returning a response to the client.
- ▶ The client application program submits a `SYNC_END_CONVERSATION` request.

For example, if the browser is closed before the conversation ends correctly, the IMS conversation is not ended explicitly and continues to exist in the system. When an IMS conversation becomes orphaned, you have no way of programmatically continuing or ending that conversation. One measure that you can take to avoid orphaned conversations is to use timeouts, such as the EJB session timeout, to force the end of a conversation that does not complete in a reasonable amount of time by submitting a `SYNC_END_CONVERSATION` request in the EJB session timeout cleanup code.

If a client application is terminated and a conversation becomes orphaned, the orphaned IMS conversation can be ended only by an IMS restart. You can check for orphaned IMS conversations in the system by running an `IMS /DISPLAY CONV` command through an `IMS_REQUEST_TYPE_IMS_COMMAND` interaction. For a list of IMS commands that are supported by OTMA, see the “Commands Supported from LU 6.2 Devices and OTMA” section in *IMS Version 12 Commands, Volume 2: IMS Commands N-V*, SC19-3010 or the “IMS Commands using OTMA” section in *IMS Version 12 Communications and Connections*, SC19-3012.

Considerations for the IMS TM Resource Adapter

IMS TMRA has some considerations regarding support of JCA 1.5 implementations.

- ▶ IMS TMRA does not provide JCA 1.5 Kerberos support.
- ▶ For a complete description of the considerations for building composite business applications that start IMS conversational transactions, see the “IMS Connect conversational support” topic in *IMS Version 12 Communications and Connections*, SC19-3012.
- ▶ New features that are added after IMS TMRA V10 and later require TCP/IP connections (no Local Option).
- ▶ For generic Java EE application servers, including WebSphere Application Server Community Edition, two-phase commit (2PC), MFS functions, and conversational and global transactions (XA) are not supported. These features are supported only when the IMS TMRA runs in WebSphere Application Server.

Important: The information that is provided here is based on the usage of WebSphere Application Server. IMS TMRA functions that are not supported in WebSphere Application Server Community Edition are not supported in other generic Java EE application servers.

A class summary for the IMS TMRA V12 Package `com.ibm.connector2.ims.ico` is shown in Table 6-2.

Table 6-2 *com.ibm.connector2.ims.ico* class summary

Class	Description
<code>IMSConnection</code>	An <code>IMSConnection</code> instance is an application-level handle that is used by a component to access an underlying physical connection to IMS Connect.
<code>IMSConnectionFactory</code>	The <code>IMSConnectionFactory</code> class provides an interface for getting connections to IMS Connect and, in turn, IMS OTMA.
<code>IMSConnectionMetaData</code>	Provides information about the Enterprise Information System (EIS) instance that is associated with a particular <code>IMSConnection</code> instance.
<code>IMSConnectionSpec</code>	An <code>IMSConnectionSpec</code> instance is used by an application component to pass information to the <code>getConnection</code> method.
<code>IMSDFSMessageException</code>	IMS TMRA throws an <code>IMSDFSMessageException</code> when IMS returns a “DFS” message and the value of the <code>imsRequestType</code> property in the <code>IMSInteractionSpec</code> instance of the interaction is 1.
<code>IMSInteraction</code>	A component uses an instance of this class to perform an interaction with IMS through IMS Connect.
<code>IMSInteractionSpec</code>	An instance of this class contains properties that are used in an interaction with IMS through IMS Connect.
<code>IMSManagedConnection</code>	An <code>IMSManagedConnection</code> is an abstract class for the IMS managed connection that represents a physical connection to IMS through IMS Connect.
<code>IMSManagedConnectionFactory</code>	An <code>IMSManagedConnectionFactory</code> instance is a factory for <code>IMSConnectionFactory</code> instances and <code>IMSManagedConnection</code> instances.
<code>IMSManagedConnectionMetaData</code>	The <code>IMSManagedConnectionMetaData</code> class provides information about a <code>ManagedConnection</code> instance and the associated Enterprise Information System (EIS).
<code>IMSResourceAdapterMetaData</code>	The <code>IMSResourceAdapterMetaData</code> class Contains meta information about the resource adapter, IMS TMRA.

Developing an application for use with the IMS TM Resource Adapter

To interact with IMS Transaction Manager (IMS TM) through the IMS TMRA, you can develop your application by using a Rational or WebSphere integrated development environment (IDE), or by coding your application outside of an IDE by using the Java Platform, Enterprise Edition Connector Architecture CCI.

An application uses the IMS TMRA to run an IMS transaction in IMS TM. The IMS TMRA also supports other types of interactions with IMS TM, such as running the IMS commands that are supported by IMS OTMA. Consider the type of interaction with IMS TM when you develop your application. In addition, the IMS TMRA implements the Java Platform, Enterprise Edition Connector Architecture CCI API, an application programming interface that you can use in your applications to communicate with an Enterprise Information System (EIS), IMS TM in this case, through the IMS TMRA.

Recommendation: When you develop your application by using an IDE, you can use the provided wizards and file importers to generate a Java EE application that runs on WebSphere Application Server. When you use an IDE to generate your code, you do not need to write the quality of service (QoS) code or the CCI code that is needed by your Java application. Instead, you can concentrate on writing code to implement your business logic and function.

Preparing to use the IMS TM Resource Adapter

Before you use the IMS TMRA, you must first determine the version of the IMS TMRA and the development environment that you need to use. To do so, complete the following steps:

1. Determine which version of the IMS TMRA you want to use. The version of the IMS TMRA that you use depends on whether you need the new features that are added to IMS TMRA and the version of your IMS.

For supported versions and software configurations, see Figure 6-18.

Version of the IMS TM resource adapter ¹	Supported version of IMS with integrated IMS Connect ²	Supported version of IBM® WebSphere® Application Server	Supported version of IBM WebSphere Transformation Extender	Supported version of IBM WebSphere Message Broker	Non-IBM Java™ EE application servers
Version 12	<ul style="list-style-type: none"> • IMS Version 12 • IMS Version 11 	<ul style="list-style-type: none"> • Version 7 or later • Version 2.1 or later for WebSphere Application Server Community Edition³ 	Version 8.4 (ships with IMS TM resource adapter Version 12.1.0)	Version 8 (ships with IMS TM resource adapter Version 12.1.0)	Any generic Java EE 1.4 or later certified application servers ³
Version 11	<ul style="list-style-type: none"> • IMS Version 12 • IMS Version 11 	<ul style="list-style-type: none"> • Version 7 or later • Version 2.1 or later for WebSphere Application Server Community Edition³ 	Version 8.3 (ships with IMS TM resource adapter Version 11.1.0)	Version 7 (ships with IMS TM resource adapter Version 10.1.1)	Any generic Java EE 1.4 or later certified application servers ³

Figure 6-18 Supported software configurations by version of the IMS TMRA

2. Determine which integrated development environment (IDE) you will use.
3. If you plan to deploy the IMS TMRA to an application server other than IBM WebSphere Application Server, you must download International Components for Unicode (ICU) Version 4.4.2 or later from the ICU website at <http://icu-project.org/download> and deploy it to your application server.
4. Review the potential migration issues and decide whether the issues apply to you.
5. Download the runtime component for the version that you choose. The runtime component is available for downloads from the IMS TMRA website. It is also available for installation on z/OS by using SMP/E.

6. If you are upgrading from an older version of the IMS TMRA, follow the steps in the “Installing IMS TMRA service and updates” topics, found at:

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.etools.ims.tmra.doc%2Ftopics%2Ftimsremove.htm>

Here are the footnotes for Figure 6-18 on page 251:

1. For the version of the IMS TMRA that you choose, see the requirements topic about supported configurations for specific functions, found at:
<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.etools.ims.tmra.doc%2Ftopics%2Ftimsremove.htm>
2. The integrated IMS Connect function can coexist with earlier or newer versions of IMS. For the version of IMS that you are using for IMS Connect coexistence considerations and coexistence APARs, see *IMS V13 Release Planning*, GC19-3658.
3. Generic Java EE 1.4 or later certified application servers, including WebSphere Application Server Community Edition, are supported if the IMS TMRA installation verification program (IVP) can run successfully. Only a subset of IMS TMRA functions is supported. For usage considerations, see the restrictions topic, found at:

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.etools.ims.tmra.doc%2Ftopics%2Ftimsremove.htm>

For generic Java EE 1.4 or later certified application servers, IBM support is provided for specific functions if either of the following items are true:

- The problem can be re-created in the supported versions of WebSphere Application Server or WebSphere Application Server Community Edition.
- The problem is confirmed to be caused by the IMS TMRA through diagnostic traces.

For a more detailed description of the compatibility of the supported architecture, development environments, and runtime environments, including the use of WebSphere Process Server and WebSphere Message Broker, go to the following website:

<http://publib.boulder.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.etools.ims.tmra.doc%2Ftopics%2Frimsssoftwareconfig.htm>

6.5.4 Issuing synchronous callout requests from a Java dependent region

IMS provides support for synchronous callout function from Java message processing (JMP) or Java batch processing (JBP) applications through an IMS implementation of the Java Message Service (JMS).

To use the JMP and JBP support for synchronous callout, the IMS Enterprise Suite JMS API `jms.jar` file must be in your class path. To download the IMS Enterprise Suite JMS API, go to the following website:

<http://www-01.ibm.com/software/data/ims/soa-integration-suite/enterprise-suite/>

The IMS implementation of JMS is limited to supporting the Point-to-Point (PTP) messaging domain only. In addition, support is provided only for non-transacted QueueSession objects with Session.AUTO_ACKNOWLEDGE mode.

If the JMP or JBP application attempts to call any JMS method that is not supported by IMS or pass any unsupported argument to JMS method calls, a `JMSEException` exception is thrown.

To send a message using the JMP and JBP support for synchronous callout and synchronously receive a response, complete the following steps:

1. Create a `com.ibm.ims.jms.IMSQueueConnectionFactory` object.
2. Create a JMS `QueueConnection` instance by calling the `createQueueConnection` method on the `IMSQueueConnectionFactory` object.
3. Create a JMS `QueueSession` instance by calling the `createQueueSession` method on the `QueueConnection` instance.

In the method call, you must set the input parameter values to false and `Session.AUTO_ACKNOWLEDGE` to specify that the generated `QueueSession` instance is non-transacted and runs in `AUTO_ACKNOWLEDGE` mode.
4. Create a queue identity by calling the `createQueue` method on the `QueueSession` instance. In the method call, you must set the input parameter value to the OTMA descriptor name for the synchronous callout operation.
5. Create a JMS `QueueRequestor` instance and pass in the `QueueSession` instance from step 3 on page 253 and the `QueueSession` instance from step 4 as input parameters to the `QueueRequestor` constructor method.
6. Create a `TextMessage` instance by calling the `createTextMessage` method on the `QueueSession` instance from step 3 on page 253. Set the string containing the message data.
7. To send the message and retrieve a response, call the request method on the `QueueRequestor` object from step 5. In the method call, pass in the `TextMessage` instance from step 6. You need to cast the return value from the request method call to a `TextMessage` instance. If the call is successful, the return value is the response to the synchronous callout request.

The code in Example 6-3 shows how to write a simple JMP or JBP application that sends a message to an external application and synchronously receive a response message. In the example, an `IMSQueueConnectionFactory` instance is created with a timeout value of 10 seconds and with 128 KB of space that is allocated to hold response messages.

Example 6-3 Sample IMSQueueConnectionFactory

```
import javax.jms.JMSException;
import javax.jms.Queue;
import javax.jms.QueueConnection;
import javax.jms.QueueRequestor;
import javax.jms.QueueSession;
import javax.jms.Session ;
import javax.jms.TextMessage;
import com.ibm.ims.jms.IMSQueueConnectionFactory;

public class IMS_Sample
{
    public static void main(String argv[])
    {
        IMSQueueConnectionFactory jmsConnectionFactory
            = new IMSQueueConnectionFactory();
        QueueConnection jmsConnection = null;
        QueueSession jmsQueueSession = null;
        Queue jmsQueue = null;
        QueueRequestor jmsQueueRequestor = null;

        try {
```

```

jmsConnectionFactory.setTimeout(1000);
    // set the timeout to 10 seconds
jmsConnectionFactory.setResponseAreaLength(128000);
    // allocate 128k to hold the response message
jmsConnection = jmsConnectionFactory.createQueueConnection();
jmsQueueSession
= jmsConnection.createQueueSession(false, Session.AUTO_ACKNOWLEDGE);
    // set session to be non-transacted and in AUTO_ACKNOWLEDGE mode
jmsQueue = jmsQueueSession.createQueue("OTMDEST1");
    // pass in the OTMA descriptor name

jmsQueueRequestor
= new QueueRequestor(jmsQueueSession, jmsQueue);
TextMessage sendMsg = jmsQueueSession.createTextMessage();
sendMsg.setText("MyMessage");
System.out.println("Sending message: "+sendMsg.getText());
TextMessage replyMsg
= (TextMessage)jmsQueueRequestor.request(sendMsg);

    System.out.println("\nReceived message: "+replyMsg.getText());
} catch (JMSException e) {
    e.printStackTrace();
}
}

```

6.5.5 IMS Enterprise Suite Connect APIs

The Connect APIs provide APIs for custom IMS Connect TCP/IP client applications to specify and configure connections and interactions with IMS Connect.

The Connect APIs are available in two languages:

- ▶ Connect API for C
- ▶ Connect API for Java

IMS Enterprise Suite Connect API for Java overview

The IMS Enterprise Suite Connect API for Java provides a simple interface for developing custom IMS Connect TCP/IP client applications that are written in Java.

IMS Connect is an optional IMS TM network component that provides high performance communication between one or more TCP/IP clients and one or more IMS systems. The Connect API for Java simplifies the process of developing Java applications that interact with IMS through IMS Connect.

Here are the key features of the Connect API for Java:

- ▶ Opens connections to IMS Connect on behalf of the client application.
- ▶ Provides client applications with programmatic control of connections.
- ▶ Assembles messages to send to IMS Connect, including the IMS request message (IRM) header.
- ▶ Manages the IMS Connect message protocol by sending and receiving the appropriate messages for interactions with IMS Connect.

The Connect API for Java manages the communication between the Java application and IMS Connect. The client application provides property values that describe the type of connection to establish with IMS Connect. The Connect API for Java establishes a connection for the application and maintains that connection while it is needed.

The Connect API for Java provides methods that encapsulate the creation of an input request message to be sent to IMS Connect and the retrieval of the resulting response from IMS Connect. The creation of the request message is based on properties that are specified in the application or loaded from a reusable properties file. The properties determine the type of interaction to be run. By completing these steps on behalf of the application, the API reduces the programming impact that is associated with the IMS Connect protocol headers.

For a synchronous callout message, the Connect API for Java stores the output message and associated correlator token for the client application in the corresponding interaction object. When the client is finished processing the message, it can return the synchronous callout response (or an error response) to the waiting IMS application program.

The Connect API for Java also provides Java application developers with flexible ways to use IMS Connect. For example, you can either acknowledge responses from IMS Connect manually in your application or have the Connect API for Java internally acknowledge responses from IMS Connect automatically.

You can use the Connect API for Java to drive IMS transactions, OTMA-supported IMS commands, CSL OM-supported IMS type-2 commands, and IMS Connect-supported commands (such as **PING** and RACF Password Change) from your Java client application. The Connect API for Java supports Secure Sockets Layer (SSL) for securing TCP/IP communications between client applications and IMS Connect.

The Connect API for Java supports the HWSSMPL0 and HWSSMPL1 IMS Connect user message exits. The default user exit is HWSSMPL1.

6.5.6 IBM IMS DB Resource Adapter (DB Universal Driver)

Here are the highlights of what was added to the Universal DB Resource Adapter for IMS V12:

- Catalog support

The Universal DB Resource Adapters can now use the IMS catalog that was released in IMS V12. This provides a trusted source of database and application metadata. By providing a central source for metadata, the ease of scaling applications is improved, as there is no file system dependency for application metadata.

- Qualify by Position Support

IMS DB has enhanced its SSA qualifications to take an offset and a length for a chunk of data in a segment instead of a field name. This completely removes the notion of “search” fields (fields that are defined directly in the IMS DBD source) as all fields can now be searched, which removes any IMS specific knowledge in forming SQL qualifications and speeds up application development and improves tool integration.

- Support for scalars

This was added for analytic purposes. Suppose that you have a value that is stored as 0 based, but you know that it is actually 1 based. You can adjust it by using **SELECT COLUMN + 1 FROM TABLE** and adjust it in the query itself instead of having to do application manipulation of the ResultSet.

- Enhanced SQL support for OLAP function

There is support for numeric functions such as SIN, COS, LOG, and EXP. This helps users write queries that produce results that they want instead of having to manipulate data afterward in an application. This is important for reporting and analytic tools, such as IBM Cognos®.

- WebLogic support

The drivers are enhanced to work with WebLogic. You can use everything that you get from using the Java EE architecture, such as connection pooling. More importantly for many customers, this also provides the capabilities to do two-phase commit.

- Performance enhancements

There are many performance enhancements to the resource adapters, such as better caching. These same technologies apply to the other Universal Drivers, which have been benchmarked at over 19000 transaction per second in a JMP region.

6.5.7 IMS Universal drivers: Configuring connections to IMS

The IMS Universal drivers can run in z/OS and distributed environments, including WebSphere Application Server for z/OS and WebSphere Application Server for distributed platforms.

The IMS Universal drivers include the following adapters and drivers:

- The IMS Universal Database resource adapters, which take advantage of Java Platform, Enterprise Edition (Java EE) services.
- The IMS Universal JDBC driver, which supports SQL calls that directly access your IMS data.
- The IMS Universal DL/I driver, which provides calls that are similar to DL/I in a Java programming interface.

When running in a distributed environment on a server such as WebSphere Application Server for distributed platforms, or in a remote z/OS environment on a server such as WebSphere Application Server for z/OS, the IMS Universal drivers connect to IMS using a type-4 connection architecture, which supports TCP/IP communications and socket management.

When running locally on the same logical partition (LPAR) as IMS, the IMS Universal drivers connect to IMS by using a type-2 connection architecture, which supports direct communication with IMS through the IMS Open Database Access (ODBA) and IMS database resource adapter (DRA) interfaces.

WebSphere Application Server supports all of the IMS Universal drivers in both distributed and z/OS environments:

- IMS Universal drivers: WebSphere Application Server type-4 connections

Java applications that run on WebSphere Application Server can access IMS databases by using the type-4 connectivity that is provided by the IMS Universal drivers. The type-4 connectivity of the IMS Universal drivers enables Java application programs to access IMS databases from a wide variety of distributed and mainframe environments, either in stand-alone mode or under an application server, with or without XA support for global transactions.

- **IMS Universal drivers: WebSphere Application Server for z/OS type-2 connections**

When WebSphere Application Server for z/OS and IMS are on the same logical partition (LPAR), Java applications running in WebSphere Application Server for z/OS can access IMS databases by using the type-2 connectivity that is provided by the IMS Universal Database resource adapters.

- **The IMS Universal drivers: CICS connections**

Java applications that run on IBM CICS Transaction Server for z/OS can access IMS databases by using the type-2 connectivity that is provided by the IMS Universal drivers. Other than the Java layer, access to IMS from a Java application is the same as for a non Java application.

Table 6-3 on page 258 shows where you use the different IMS Universal Database resource adapters that you need. In your z/OS UNIX file system on the IMS mount point, this location is usually under **usr** → **lpp** → **ims**. Figure 6-19 shows where your mount point might be different.

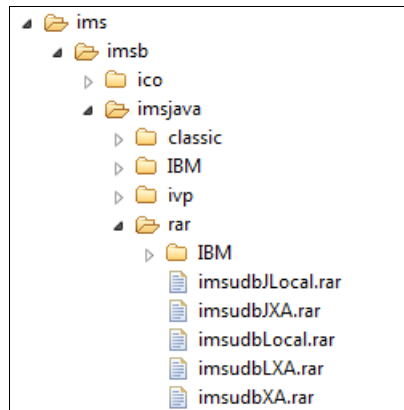


Figure 6-19 z/OS UNIX mount point sample

Table 6-3 IMS Universal resource adapter comparison

Application platform	Data access method	Transaction processing that is required	Recommended approach
WebSphere Application Server for distributed platforms or WebSphere Application Server for z/OS	JDBC programming interface to perform SQL data operations	Local transaction processing only	Use the IMS Universal JCA/JDBC driver version of the IMS Universal Database resource adapter with local transaction support (imsudbJLocal.rar), and make SQL calls with the JDBC API.
	JDBC programming interface to perform SQL data operations	Two-phase (XA) commit processing ^a or local transaction processing	Use the IMS Universal JCA/JDBC driver version of the IMS Universal Database resource adapter with XA transaction support (imsudbJXA.rar), and make SQL calls with the JDBC API.
	JDBC programming interface to perform SQL data operations	Two-phase (XA) commit processing ^b or local transaction processing	Use the IMS Universal JDBC driver (imsudb.jar), and make SQL calls with the JDBC API.
	CCI programming interface to perform SQL or DL/I data operations	Local transaction processing only	Use the IMS Universal Database resource adapter with local transaction support (imsudbLocal.rar), and make SQL calls with the SQLInteractionSpec class or DL/I calls with the DLInteractionSpec class.
	CCI programming interface to perform SQL or DL/I data operations	Two-phase (XA) commit processing ^a or local transaction processing	Use the IMS Universal Database resource adapter with XA transaction support (imsudbXA.rar), and make SQL calls with the SQLInteractionSpec class or DL/I calls with the DLInteractionSpec class.
Stand-alone Java application (outside a Java EE application server) that is on a distributed platform or a z/OS platform	Traditional DL/I programming semantics to perform data operations	Two-phase (XA) commit processing ^b or local transaction processing	Use the IMS Universal DL/I driver (imsudb.jar), and make DL/I calls with the PCB class.
Stand-alone non-Java application that is on a distributed platform or a z/OS platform	Data access using DRDA protocol	Two-phase (XA) commit processing or local transaction processing	Use a programming language of your choice to issue DDM commands to IMS Connect. The application programmer is responsible for implementing the two-phase commit mechanism.

a. XA transaction support is available only with type-4 connectivity.

- b. The driver is enabled for local and XA transactions, but the application programmer is responsible for implementing the two-phase commit mechanism. XA transaction support is available only with type-4 connectivity.

There is support for type-2 and type-4 connections with the Universal DB resource adapters.

► Distributed access

Universal drivers support type-4 connectivity to IMS databases from the following TCP/IP enabled platforms and run times:

- Windows
- Linux on System z
- z/OS
- WebSphere Application Server
- Stand-alone Java SE

Resource Recovery Services (RRS) is *not* required if applications do not require distributed two phase commit.

The Universal Database drivers have a framework that can process any of the three main programming models: Java Platform, Enterprise Edition, JDBC, and DLI. The Universal Database drivers can connect to any IMS subsystem on any mainframe system. The same application can have active connections to any number of IMS systems on any number of mainframe installations.

There is type-2 access (IMS access from the same LPAR in WebSphere Application Server z/OS, IMS JDR, CICS, or DB2 for z/OS stored procedures) using these same Universal Database drivers, as shown in Figure 6-20. Again, the same framework can handle both type-2 and type-4 connectivity, so the applications themselves do not change (only the connection properties).

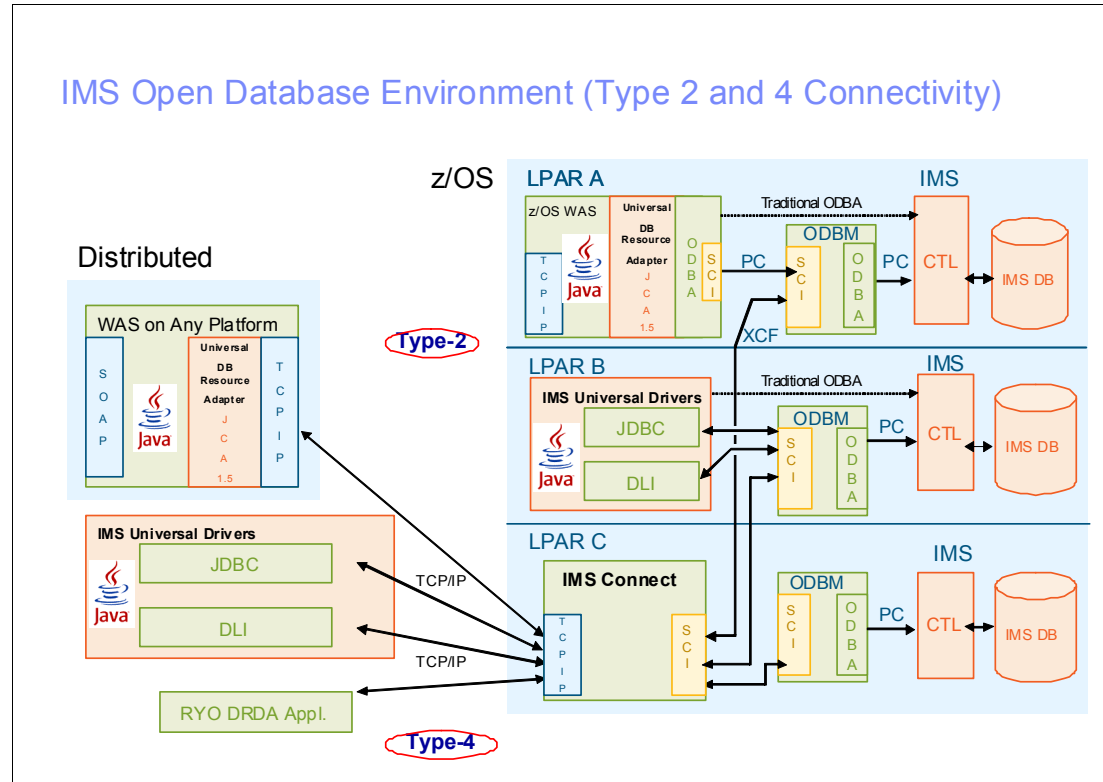


Figure 6-20 Type 2 and Type 4 Connectivity

For more details and programming information, go to the IMS information center found at:

http://pic.dhe.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.ims12.doc%2Fimshome_v12.htm



Part 2

Extended architecture and application technology of IBM IMS

IBM IMS connectivity enables clients to easily integrate IMS across the enterprise: trusted IMS applications and transactions may become web services. New points of integration for IMS across the enterprise offer capabilities such as database visualization, query tools support, data insight, open access, SQL and .NET support, analytics, and so, which contribute to the modernization of IMS IT infrastructures.

This part describes samples of the enhanced application technologies that allow the repurposing of IMS data assets as web services, mobile apps, and so on, thus preserving and extending the return on investment (ROI) of the original investment.

This part includes the following chapters:

- ▶ Chapter 7, “COBOL dynamic SQL access to IBM IMS databases” on page 263
- ▶ Chapter 8, “The IMS catalog” on page 273
- ▶ Chapter 9, “IMS Explorer and Open Database connectivity” on page 347
- ▶ Chapter 10, “IMS Data Provider for Microsoft .NET” on page 367
- ▶ Chapter 11, “IMS mobile enablement solutions” on page 383
- ▶ Chapter 12, “IBM WebSphere DataPower and IMS integration” on page 395



COBOL dynamic SQL access to IBM IMS databases

Native SQL support for COBOL enables SQL in COBOL programs to access IMS databases and provides for SQL processing natively in IMS. This support, available in IMS V13, is in addition to the Java based SQL IMS application support.

Native SQL support for COBOL allows SQL programmers to access IMS databases using SQL statements such as the ones that are issued for relational databases. This enhancement expands IMS database usage to a wider group of application and database developers and can take advantage of SQL skills without requiring in-depth IMS database knowledge.

This function requires IBM Enterprise COBOL for z/OS V5.1 and APAR PM92523.

This chapter briefly describes an example of SQL support for COBOL by referencing a simple COBOL program.

This chapter covers the following topics:

- ▶ Overview
- ▶ SQL statements that are supported by IMS
- ▶ Sample IMS COBOL SQL program

7.1 Overview

Over recent releases, the information in your IMS databases has been made accessible through an increasing number of avenues. Initially, the only access was through host applications running on z/OS, using Data Language 1. More recently, distributed access to IMS data has been possible through Java applications using the JDBC interface.

IMS V13 added a dimension to access and update the information in your IMS databases. The existing Data Language 1 (DL/I) interface continues to be available and supported for all your applications. When using Enterprise COBOL Version 5, applications can now access and update information in these same IMS databases by using the Structured Query Language (SQL).

Using the extended database descriptions through the catalog function in IMS V12, these COBOL SQL programs can access IMS databases by using the relational column and table paradigm. The SQL that is supported by IMS is a subset of the full SQL language, so application programming skills gained through use of DB2 for z/OS and other relational database management systems are immediately usable in the IMS SQL environment. In this release, IMS provides support for dynamic SQL.

IMS makes no distinction between applications accessing IMS databases through DL/I, and applications accessing IMS databases through SQL. These IMS SQL applications can inquire, and update, the same set of IMS databases in the same manner as all existing IMS applications. The DL/I and SQL interfaces can be used in the same program, and can even access the same databases in the program. These applications can run as Message Processing Programs (MPPs), Interactive Fast Path (IFP) programs, or Batch Message Processing (BMP) programs. The IMS databases are accessible through a full IMS TM/DB system, or through IMS DBCTL. Currently, IMS SQL access is not possible with CICS applications or DL/I Batch applications.

IMS SQL uses the same environment to access IMS databases as native DL/I applications. The database is defined through the standard data base description (DBD), and program access is provided through the standard program specification block (PSB). The DBD and PSB are used to produce the standard access control block (ACB). The IMS catalog must be populated, either by using the combined ACB Generation and Catalog Populate utility (DFS3UACB), or the two utilities separately, that is, the ACBGEN utility and the IMS Catalog Populate utility (DFS3PU00).

IMS SQL applications require the IMS catalog for the definition of all tables (segments) and columns (fields). The catalog is accessed at run time for these definitions.

7.2 SQL statements that are supported by IMS

Here are the SQL statements that are supported by IMS:

CLOSE	The CLOSE statement closes a cursor.
DECLARE CURSOR	The DECLARE CURSOR statement defines a cursor.
DECLARE STATEMENT	The DECLARE STATEMENT statement is used for application program documentation. It declares names that are used to identify prepared SQL statements.
DELETE	The DELETE statement deletes rows from a table.

DESCRIBE OUTPUT	The DESCRIBE OUTPUT statement obtains information about a prepared statement.
EXECUTE	The EXECUTE statement runs a prepared SQL statement.
FETCH	The FETCH statement positions a cursor on a row of its result table. It can return zero or one and assigns the values of the rows to host variables if there is a target specification.
INCLUDE	The INCLUDE statement inserts application code, including declarations and statements, into a source program.
INSERT	The INSERT statement inserts rows into a table.
OPEN	The OPEN statement opens a cursor so that it can be used to process rows from its result table.
PREPARE	The PREPARE statement creates an executable SQL statement from a string form of the statement. The character-string form is called a statement string. The executable form is called a prepared statement.
SELECT	The SELECT statement is used to retrieve data from one or more tables. The result is returned in a tabular result set.
UPDATE	The UPDATE statement updates the values of specified columns in rows of a table.
WHENEVER	The WHENEVER statement specifies the host language statement to be run when a specified exception condition occurs.

7.3 Sample IMS COBOL SQL program

Example 7-1 shows a sample IMS COBOL SQL program. The program is also available for download as described in Appendix B, “Additional material” on page 461.

Example 7-1 Sample IMS COBOL SQL program

```

_CBL SIZE(4000K) SOURCE XREF
_CBL SQLIMS
IDENTIFICATION DIVISION.
PROGRAM-ID. 'sqla12'.
*****
*                                     *
*   Read information from the LGI (IMS) database using SQL   *
*                                     *
*   Program          SQLA12                                     *
*   PSB:             S2U1PSB                                     *
*   Databases:       S2U1DBD                                     *
*   Access:          SQLIMS                                     *
*                                     *
*****
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  IBM-zSeries.
OBJECT-COMPUTER.  IBM-zSeries.
DATA DIVISION.
WORKING-STORAGE SECTION.
*****
*   These are the SSAs used to retrieve the database   *
*

```

```

*   and PSB information from the catalog                               *
*   This is done with the show-catalog procedure                       *
*****
01 dbd-ssa.
    02 filler          pic x(8)  value 'HEADER '.
    02 filler          pic x(1)  value '('.
    02 ssa-field-name  pic x(8)  value 'RHDRSEQ '.
    02 ssa-boolean     pic x(2)  value '= '.
    02 ssa-field-value pic x(8)  value 'DBD    '.
    02 ssa-dbd-name    pic x(8) .
    02 filler          pic x(1)  value ')'.
01 psb-ssa.
    02 filler          pic x(8)  value 'HEADER '.
    02 filler          pic x(1)  value '('.
    02 ssa-field-name  pic x(8)  value 'RHDRSEQ '.
    02 ssa-boolean     pic x(2)  value '= '.
    02 ssa-field-value pic x(8)  value 'PSB    '.
    02 ssa-psb-name    pic x(8) .
    02 filler          pic x(1)  value ')'.
77 gur-call           pic x(4)  value 'GUR '.
77 sync-call          pic x(4)  value 'SYNC'.
01 gur-returned-data.
    03 gur-header      pic x(56).
    03 gur-information pic x(15000).
*****
*   This program uses the AIB interface, since the GUR call
*   to retrieve information from the catalog requires the
*   use of the AIB interface. The program could also use
*   CBLTDLI calls if desired, when accessing IMS databases
*****
01 AIB.
    02 AIBRID          PIC x(8).
    02 AIBRLN          PIC 9(9) USAGE BINARY.
    02 AIBRSFUNC       PIC x(8).
    02 AIBRSNM1        PIC x(8).
    02 AIBRSNM2        PIC x(8).
    02 AIBRESV1        PIC x(8).
    02 AIBOALEN        PIC 9(9) USAGE BINARY.
    02 AIBOAUSE        PIC 9(9) USAGE BINARY.
    02 AIBRESV2        PIC x(12).
    02 AIBRETRN        PIC 9(9) USAGE BINARY.
    02 AIBREASN        PIC 9(9) USAGE BINARY.
    02 AIBERRXT        PIC 9(9) USAGE BINARY.
    02 AIBRESA1        USAGE POINTER.
    02 AIBRESA2        USAGE POINTER.
    02 AIBRESA3        USAGE POINTER.
    02 AIBRESV4        PIC x(40).
    02 AIBRSARE        OCCURS 18 TIMES USAGE POINTER.
    02 AIBRTOKN        OCCURS 6 TIMES  USAGE POINTER.
    02 AIBRTOKC        PIC x(16).
    02 AIBRTOKV        PIC x(16).
    02 AIBRTOKA        OCCURS 2 TIMES PIC 9(9) USAGE BINARY.
*   Data Input Area from the terminal
*   (not used in this program).
01 INPUT-MESSAGE.

```

```

03 IN-LL      PIC S9(4) COMP.
03 IN-ZZ      PIC S9(4) COMP.
03 IN-TRANCOD PIC X(8).
03 IN-DATA.
04 IN-LINE1   PIC X(80).
* Data Output Area from the terminal
* (not used in this program).
01 OUTPUT-AREA.
02 OUT-LL     PICTURE S9(3) COMP VALUE +100.
02 OUT-ZZ     PICTURE S9(3) COMP VALUE +0.
02 OUT-LINE   PICTURE X(96) VALUE SPACES.
02 OUT-DATA   REDEFINES OUT-LINE.
04 OUT-TEXT   PIC X(32).
04 OUT-MSG1   PIC X(32).
04 OUT-MSG2   PIC X(32).
*****
* SQL STRING *
* In this program, the SQL statement is passed to IMS
* via a working storage text variable. The first two
* bytes are the length of the SQL statement, and the
* rest is the statement itself.
*****
01 FULL-STATEMENT.
05 SELECT-STATEMENT.
09 SELECT-STATEMENT-LEN PIC S9(4) COMP VALUE +200.
09 STATEMENT-TXT       PIC X(200).
* Miscellaneous fields for this program
77 row-count          PIC 9(6) VALUE 0.
77 loop-count         PIC 9(6) VALUE 0.
*****
* SQL INCLUDE FOR SQLCA *
* IMS SQL Communications Area *
*****
EXEC SQLIMS
INCLUDE SQLIMSCA
END-EXEC.
*****
* SQL INCLUDE FOR SQLCA *
* IMS SQL Descriptor Area *
*****
*
* it is retrieved, and displayed for your interest,
* but not used in this program
EXEC SQLIMS
INCLUDE SQLIMSDA
END-EXEC.
*****
* SQLCODE TEMPLATE TO REMOVE SIGN BIT. *
*****
77 SQLIMSCODE-DISP PIC S9(9) DISPLAY SIGN LEADING SEPARATE.
*****
* district SEGMENT *
*****
01 sql-district.
03 sql-distnum   pic x(6).

```

```

03 sql-distname    pic x(30).
03 sql-currdate    pic x(6).
03 sql-currtime    pic x(8).
03 sql-custno      pic x(6).
03 sql-distrest    pic x(5).
*****
LINKAGE SECTION.
01 IOPCB.
02 IO-LTERM        PIC X(8).
02 IO-RESV         PIC X(2).
02 IO-STATUS       PIC X(2).
02 IO-PREF         PIC X(12).
02 IO-MODN         PIC X(8).
02 IO-USERID       PIC X(8).
02 IO-GROUPID      PIC X(8).
01 DBPCB.
02 DBNAME          PICTURE X(8).
02 SEG-LEVEL-NO    PICTURE X(2).
02 DBSTATUS        PICTURE XX.
02 FILLER          PICTURE X(2000).
PROCEDURE DIVISION USING IOPCB, DBPCB.
MAIN-Routine.
    DISPLAY "COBOL Program sqla12 Execution begins... ".
    EXEC SQLIMS
        WHENEVER SQLERROR GOTO 100-DBERROR
    END-EXEC.
    EXEC SQLIMS
        WHENEVER SQLWARNING GOTO 200-DBERROR
    END-EXEC.
    EXEC SQLIMS
        WHENEVER NOT FOUND CONTINUE
    END-EXEC.
    perform show-catalog through show-catalog-end .
    display "about to process DECLARE TELE1 CURSOR FOR DYSQL".
    EXEC SQLIMS
        DECLARE TELE1 CURSOR FOR DYSQL
    END-EXEC.
    MOVE ZERO to LOOP-COUNT.
    MOVE "SELECT distnum, distname, currdate, currtime,
-      "custno, distrest FROM district
-      "where distnum = '000010'"
      TO STATEMENT-TXT.
    move 198 to SELECT-STATEMENT-LEN.
    display "about to process PREPARE DYSQL FROM :SELECT-".
    EXEC SQLIMS
        PREPARE DYSQL FROM :FULL-STATEMENT
    END-EXEC.
    move 6 to SQLIMSn.
    exec SQLIMS
        describe output dysql into :sqlimsda
    end-exec.
    perform display-sqlda.
    display "about to process OPEN TELE1".
    EXEC SQLIMS
        OPEN TELE1

```

```

END-EXEC.
PERFORM FETCH-DISTRICT
    UNTIL SQLIMSCODE NOT EQUAL 0
    display "about to perform CLOSE TELE1".
EXEC SQLIMS
    CLOSE TELE1
END-EXEC.
PERFORM sync-and-return.
FETCH-DISTRICT.
EXEC SQLIMS
    FETCH TELE1 INTO :sql-distnum, :sql-distname,
                    :sql-currrdate, :sql-currrtime,
                    :sql-custno, :sql-distrest
END-EXEC.
IF SQLIMSCODE EQUAL 0
    ADD 1 TO row-count
    display " count ", row-count,
           "| sql-distnum |", sql-distnum,
           "| sql-distname |", sql-distname,
           '|', sql-currrdate,
           '|', sql-currrtime,
           '|', sql-custno,
           '|', sql-distrest

    end-if.
*****
*   Display 'Data Not Found' Message when data not found.
*****
DATA-NOT-FOUND.
    DISPLAY " -----".
    DISPLAY " ".
    DISPLAY "*****NO MORE DATA FOUND*****".
*   MOVE SQLIMSCODE TO SQLIMSCODE-DISP.
*   DISPLAY "SQLIMSCODE = " SQLIMSCODE-DISP.
*
*****
*   Display error message
*****
100-DBERROR.
    DISPLAY "ERROR OCCURRED IN WHENEVER SQLIMSError".
    MOVE SQLIMSCODE TO SQLIMSCODE-DISP.
    DISPLAY "SQLIMSCODE = " SQLIMSCODE-DISP.
    DISPLAY "SQLIMSERRMC= " SQLIMSERRMC.
    DISPLAY "SQLIMSSTATE= " SQLIMSSTATE.
    perform sync-and-return.
*****
*   Display warning message
*****
200-DBERROR.
    DISPLAY "ERROR OCCURRED IN WHENEVER SQLWARNING".
    MOVE SQLIMSCODE TO SQLIMSCODE-DISP.
    DISPLAY "SQLIMSCODE = " SQLIMSCODE-DISP.
    DISPLAY "SQLIMSERRMC= " SQLIMSERRMC.
    DISPLAY "SQLIMSSTATE= " SQLIMSSTATE.
    perform sync-and-return.
*****

```

```

*      Time to go home again...
*****
sync-and-return.
    DISPLAY "Returning ...".
    call 'CBLTDLI' using sync-call iopcb.
    GOBACK.
*****
*
*      This routine accesses the IMS catalog information with
*      the Get Unique Record (GUR) call through an
*      Application Interface Block (AIB)
*
*      Applications can use the CBLTDLI, AIB and now the
*      SQL interface to IMS databases in the same program,
*      even to the same database (PCB) if needed.
*
*****
show-catalog.
    move "C1CSTP "      to ssa-dbd-name.
    move "C1ACSTLD"     to ssa-psb-name.
*      Retrieve the DBD information from the catalog
    move "DFSAIB "      to AIBRID.
    move length of aib to AIBRLen.
    move length of gur-returned-data to AIBOALEN.
    move spaces         to AIBRSFUNC.
    MOVE "DFSCAT00"     TO AIBRSNM1.
    call 'AIBTDLI' using gur-call
                        aib
                        gur-returned-data
                        dbd-ssa.
    set address of dbpcb to aibresal.
    if dbstatus not equal space
        display '|' dbstatus '|,' 'aib return is ', AIBRETRN,
            ',aib reason is ' AIBREASN
    else
        display "----- D B D -----".
        display gur-information .
*      Now retrieve the PSB information from the catalog
    call 'AIBTDLI' using gur-call aib
                        gur-returned-data psb-ssa.
    if dbstatus not equal space
        display '|' dbstatus '|,' 'aib return is ', AIBRETRN,
            ',aib reason is ' AIBREASN
    else
        display " ".
        display "----- P S B -----".
        display gur-information .
show-catalog-end.
    exit.
*****
*
*      This routine displays the information returned
*      from the DESCRIBE STATEMENT request into the
*      SQL IMS Descriptor Area (SQLIMSDA).
*

```



```

*****
display-sqlda.
  display "SQLIMSDAID|" SQLIMSDaid
         "SQLIMSDABC|" SQLIMSDabc
         "SQLIMSN|" SQLIMSn
         "SQLIMSD|" SQLIMSD.
  perform with test after
    varying LOOP-COUNT from 1 by 1
    until loop-count > SQLIMSD
    display
      " ",      loop-count,
      " type=",  SQLIMSTYPE(loop-count),
      " len=" ,  SQLIMSLEN(loop-count),
      " name1=", SQLIMSNAM1(loop-count),
      " namec="  SQLIMSNAMC(loop-count),
      "|"
    end-perform.
END PROGRAM "sqla12".

```



The IMS catalog

This chapter provides a description of the IBM IMS catalog and the way it can be used to expand and consolidate the information about IMS databases and their metadata.

The IMS catalog is an optional system database (available since IMS 12) that stores metadata about your databases and applications. Its comprehensive view of IMS database metadata, fully managed by IMS, allows IMS to participate in solutions that require the exchange of metadata. A type of solution that requires such an exchange is business impact analysis.

The IMS Universal drivers are enhanced to take advantage of the IMS catalog.

This chapter covers the following topics:

- ▶ Overview and objectives of the catalog
- ▶ Physical structure of the catalog database
- ▶ IMS catalog database installation and management
- ▶ Application usage of the catalog
- ▶ The role of the IMS Enterprise Suite Explorer for Development
- ▶ Using IMS Explorer to capture IMS metadata
- ▶ Enhancements to the IMS Universal drivers

This chapter describes the IMS catalog, a trusted online source for both IBM IMS database and application metadata information, and its usage by IMS Open Database and IMS Explorer. This chapter targets application developers who can benefit from these simplification and integration functions when they are facing any large-scale deployment of the IMS Open Database solution.

8.1 Overview and objectives of the catalog

Before the introduction of the IMS catalog, information about the structure of the Data Language/I (DL/I) databases was spread across the following different data structures:

- The database description

The *database description* (DBD) defines the characteristics of a database. For example, the DBD defines characteristics such as its organization and access method, the segments and fields in a database record, and the relationship between the types of segments. Usually, information for the segments is limited to the few fields that are required to identify and search for segments across the hierarchical structures, that is, limited to the fields that are used as search arguments in the segment search argument (SSA), and the fields that are required to define logical relationships and secondary indexes.

- The COBOL copybooks and PL/I or C include members

The details of the fields in each segment of the database are often defined in a Common Business Oriented Language (COBOL) copybook and in PL/I or C include members. These members detail all the fields in each database segment (not just the fields that are used in SSAs), and are included into the source program when the program is compiled.

- The program specification block

A *program specification block* (PSB) is used to define two things: The view of the databases to be used by a program, and any logical message destinations. These views are called *program communication blocks* (PCBs) because they are the means of communicating between the application program and IMS. There can be many PCBs in a PSB (as shown in Figure 8-1), allowing a program to communicate with (access) multiple IMS databases. For each database the program can access, a PCB is needed that describes the segments and the hierarchical structure that the program can access. The description can also indicate the sensitivity that a program has to the information, that is, whether the program can see only a subset of the segment types in the database, and a subset of the fields in these segments.

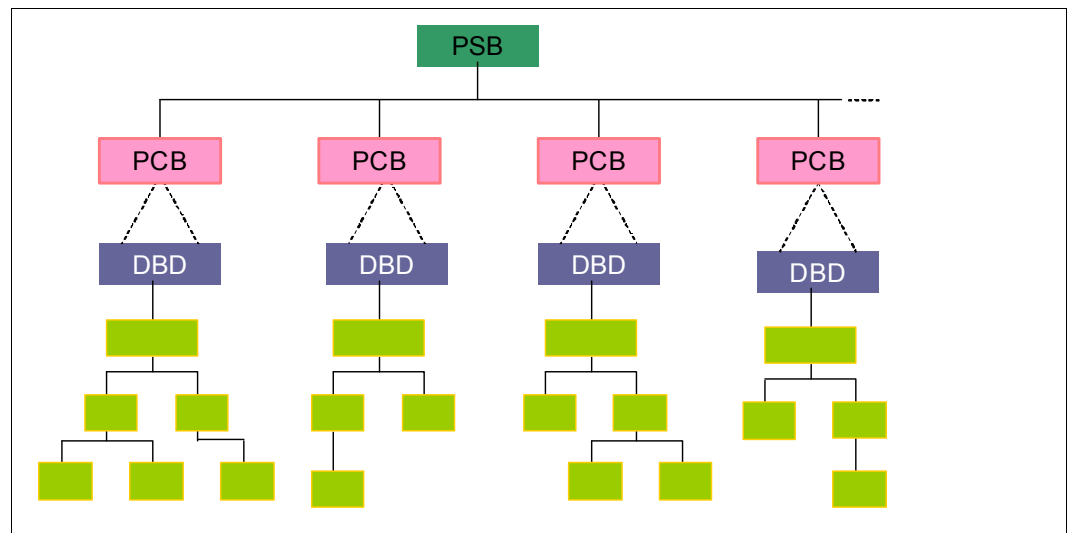


Figure 8-1 A PSB with multiple PCBs

A PCB can also allow a program to use different access paths through a database. It can allow the program to access a database through a secondary index or a logical relationship. The program view of the hierarchical structure of the database can be different from the hierarchical structure that is defined in the DBD, as shown in Figure 8-1 on page 274.

For online IMS systems, *application control block* (ACB) libraries contain the metadata that is used by IMS, which is created by performing an ACB generation from the DBD and PSB libraries. The system that uses DBD, PSB, and ACB libraries is proprietary. Typically, a DBD defines only a subset of the fields in a database record, such as the sequence (key) fields and any fields that are needed by a secondary index. Other fields in a record are typically defined only in a COBOL copybook or in a PL/I or C include file that is used by an application program.

The introduction of Java access to IMS data from JBP or JMP regions, or from remote systems or clients, requires easily accessible metadata. To provide this access, the contents of ACBLIB are offered as a Java class.

With the move to making IMS data more widely and easily accessible outside of the mainframe, application programmers must have easy and consistent access to the metadata. Since IMS V8, this data is provided by the DLIMODEL utility. This utility generates Java class files, containing a static definition of the database design at the point when the DLIMODEL utility is run.

The drawback of using the DLIMODEL utility is that it can be a challenge to manage. Also, change control is needed when the underlying database definition changes. This method potentially allows the creation of multiple copies of the database metadata, each one of which must be updated with any database structure change.

There is a requirement for a source of metadata that is easier to manage and can be trusted to reflect the possible changes of the design of IMS databases. The metadata can also be defined in an open format.

IMS 12 introduced the IMS catalog. The IMS catalog contains the metadata for databases and PSBs. It is accessed by using the Java Database Connectivity (JDBC) drivers and is also available to any tool or application. IMS metadata is available in an Extensible Markup Language (XML) format, and also available to standard DL/I applications in a traditional IMS segment format.

When the metadata is updated, the catalog is also updated to reflect the change; therefore, the data is always current. The data is current because the metadata is accessed dynamically from the active IMS catalog *high availability large database* (HALDB), rather than from static class files.

The catalog in IMS also includes a versioning system. This system allows the current version of the metadata, and a user-specified number of previous versions, to be kept and available for the applications.

The IMS catalog is a *partitioned hierarchical indexed direct access method* (PHIDAM) database that contains trusted metadata for IMS databases and applications. The *IMS Catalog Populate* utility (DFS3PU00) can optionally be used to initially populate the catalog by reading an existing ACBLIB and loading the available metadata into the catalog. All the information that is contained in the ACBLIB run time (which was derived from the DBDs and PSBs) is available to the users in the IMS catalog. Enhancements to the existing DBDGEN, PSBGEN, and ACBGEN processes allow database and application metadata to be defined to IMS.

The new *ACB Generation and Catalog Populate* utility (DFS3UACB) automatically updates the IMS catalog when the ACB members are generated. This update keeps the catalog always in a trusted state after initialization. The IMS catalog is the single, authoritative source of database and application metadata for all client applications. The catalog can also contain application metadata, such as decimal data specifications (scale and precision), data structure definitions, and mapping information.

The IMS catalog uses time stamps to identify the version of the metadata for each database, and can store multiple versions of metadata for each database. IMS provides the facility to keep a number of generations of this metadata, and remove old metadata, according to a maximum number of generations or longevity that you specify. By default, the IMS catalog contains information for each DBD and PSB in the active IMS system ACBLIB. The IMS Catalog Record Purge utility (DFS3PU10) must be run to remove the extra definitions in the catalog.

8.2 Physical structure of the catalog database

The IMS catalog is a HALDB database that is a partitioned, IMS full-function database type. Before loading records into an IMS catalog, you must define the partitions to IMS. The IMS catalog is composed of the following objects:

- ▶ Primary database DFSCD000

The access method is *overflow sequential access method* (OSAM), composed of the following database data sets:

- Primary index data set
- Indirect list data set (ILDS)
- Four data set groups for the segments of the IMS catalog records

- ▶ Secondary index DFSCX000

This index provides a cross-reference between the PSB segment and the DBD it is used to access: The data set for the secondary index database is a partitioned secondary index (PSINDEX).

The data that is stored in the IMS catalog includes all the metadata that has been traditionally held in the DBD and PSB libraries. The IMS catalog also includes extra metadata information that is not available in the earlier releases of the DBD and PSB libraries. The IMS Explorer for Development is used to extend the metadata in the DBD, which is stored in the catalog. For details about the process to include the extended metadata into the DBD and the IMS catalog, see 8.5, “The role of the IMS Enterprise Suite Explorer for Development” on page 304.

8.2.1 Segments of the catalog database

The following segments are part of the catalog:

- ▶ Resource header (item name and type) for each DBD and PSB
- ▶ DBD resources:
 - Database structure definitions (ACCESS, RMNAME, and so on)
 - Data capture parameters
 - Physical database data set (DATASET) or area (AREA) definitions
 - Segment definitions (SEGM)
 - Field definitions (FIELD)
 - Marshaller definitions (DFSMARSH)

- ▶ PSB resources:
 - Program Communication Block (PCB)
 - Sensitive segments (SENSEG)
 - Sensitive fields (SENFLD)
 - DBD cross-reference

For each macro that used in a DBD or PSB, there is an equivalent segment in the IMS catalog database.

The segments in the DBD of the IMS catalog database are shown in Example 8-1.

Example 8-1 Segments in the IMS catalog database (DFSCD000)

															DDDD	FFFF	SSS	CCC	DDDD	000	000	000																					
															D	D	F	S	S	C	C	D	D	0	0	0	0	0	0														
															D	D	F	S		C		D	D	0	00	0	00	0	00														
															D	D	FFF		SSS	C		D	D	0	0	0	0	0	0	0	0	0	0										
															D	D	F			S	C	D	D	00	0	00	0	00	0														
															D	D	F	S	S	C	C	D	D	0	0	0	0	0	0	0													
															DDDD	F			SSS	CCC	DDDD		000		000		000																
VERSION=04/18/12 11.49																																											
LEVELS= 8 SEGMENTS= 62 DATA SET GROUPS= 4																																											
ID=	1*	N/A			PRIME DS LOG. RCD. LEN=	0,	BLOCKSIZE=	32752											SEGMENT LENGTH MAX=	614,	MIN=	126																					
					OFLW DS LOG. RCD. LEN=		BLOCKSIZE=	32752											KEY LENGTH MAX=	17,	MIN=	16	NUMBSEG=	3																			
ID=	2+	N/A			PRIME DS LOG. RCD. LEN=	0,	BLOCKSIZE=	32752											SEGMENT LENGTH MAX=	934,	MIN=	66																					
					OFLW DS LOG. RCD. LEN=	0,	BLOCKSIZE=	32752											KEY LENGTH MAX=	2,	MIN=	2	NUMBSEG=	11																			
ID=	3"	N/A			PRIME DS LOG. RCD. LEN=	0,	BLOCKSIZE=	32752											SEGMENT LENGTH MAX=	934,	MIN=	326																					
					OFLW DS LOG. RCD. LEN=	0,	BLOCKSIZE=	32752											KEY LENGTH MAX=	2,	MIN=	2	NUMBSEG=	3																			
ID=	4.	N/A			PRIME DS LOG. RCD. LEN=	0,	BLOCKSIZE=	32752											SEGMENT LENGTH MAX=	4022,	MIN=	46																					
					OFLW DS LOG. RCD. LEN=	0,	BLOCKSIZE=	32752											KEY LENGTH MAX=	17,	MIN=	2	NUMBSEG=	45																			
SEG-NAME	SC#	LV	PAR	-LEN-	---	FREQ---	C	P	P	L	L	P	P	P	L	L	E	RULES	PHYS.																								
							T	T	P	T	P	H	C	C	C	C	P		N-SEQ	SEG-NAME	D-B-NAME	FORM	LOG	CHLD																			
							R	F	B		F	B	F	B	F	.L	.F	.L	S	I	D	R	INSRT	OR	OR	OR	INSRT																
HEADER	*	1	1	0	56	0.00	XX									13			P	P	P	LAST					VAR	LEN															
																									PFX	LEN=	70	MAX=	56	PFX+MAX=	126												
																												MIN=	24	PFX+MIN=	94												
DBD	*	2	2	1	552	0.00	XX	X					10						P	P	P	LAST					VAR	LEN															
																									PFX	LEN=	62	MAX=	552	PFX+MAX=	614												
																												MIN=	505	PFX+MIN=	567												
CAPXDBD	.	3	3	2	32	0.00	X	X											P	P	P	LAST					VAR	LEN															
																									PFX	LEN=	18	MAX=	32	PFX+MAX=	50												
																												MIN=	28	PFX+MIN=	46												
DBDRMK	.	4	3	2	264	0.00	XX	X											P	P	P	LAST					VAR	LEN															
																									PFX	LEN=	22	MAX=	264	PFX+MAX=	286												
																												MIN=	20	PFX+MIN=	42												
DSET	.	5	3	2	96	0.00	XX	X					1						P	P	P	LAST					VAR	LEN															
																									PFX	LEN=	26	MAX=	96	PFX+MAX=	122												
																												MIN=	70	PFX+MIN=	96												
DSETRMK	.	6	4	5	264	0.00	XX	X											P	P	P	LAST					VAR	LEN															
																									PFX	LEN=	22	MAX=	264	PFX+MAX=	286												
																												MIN=	20	PFX+MIN=	42												
AREA	.	7	3	2	40	0.00	XX	X					1						P	P	P	LAST					VAR	LEN															
																									PFX	LEN=	26	MAX=	40	PFX+MAX=	66												
																												MIN=	26	PFX+MIN=	52												
AREARMK	.	8	4	7	264	0.00	XX	X											P	P	P	LAST					VAR	LEN															
																									PFX	LEN=	22	MAX=	264	PFX+MAX=	286												
																												MIN=	20	PFX+MIN=	42												
SEGM	+	9	3	2	376	0.00	XX	X					5						P	P	P	LAST					VAR	LEN															
																									PFX	LEN=	42	MAX=	376	PFX+MAX=	418												
																												MIN=	341	PFX+MIN=	383												
CAPXSEGM	.	10	4	9	32	0.00	X	X											P	P	P	LAST					VAR	LEN															

278 IMS Integration and Connectivity Across the Enterprise

The hierarchical structure of the IMS catalog database is shown in Figure 8-2. Some segment types were omitted for display reasons, but most of the important ones are shown. Example 8-1 on page 277 shows the complete list. To see the complete picture of the catalog database, import the catalog database DBD (DFSCD001) into IMS Explorer. The source of this DBD is available in the SDFSSRC data set, which has been delivered with IMS since Version 12. Instructions for importing DBDs into IMS Explorer are described in 8.6, “Using IMS Explorer to capture IMS metadata” on page 307.

Figure 8-2 shows the database structure for the IMS catalog database.

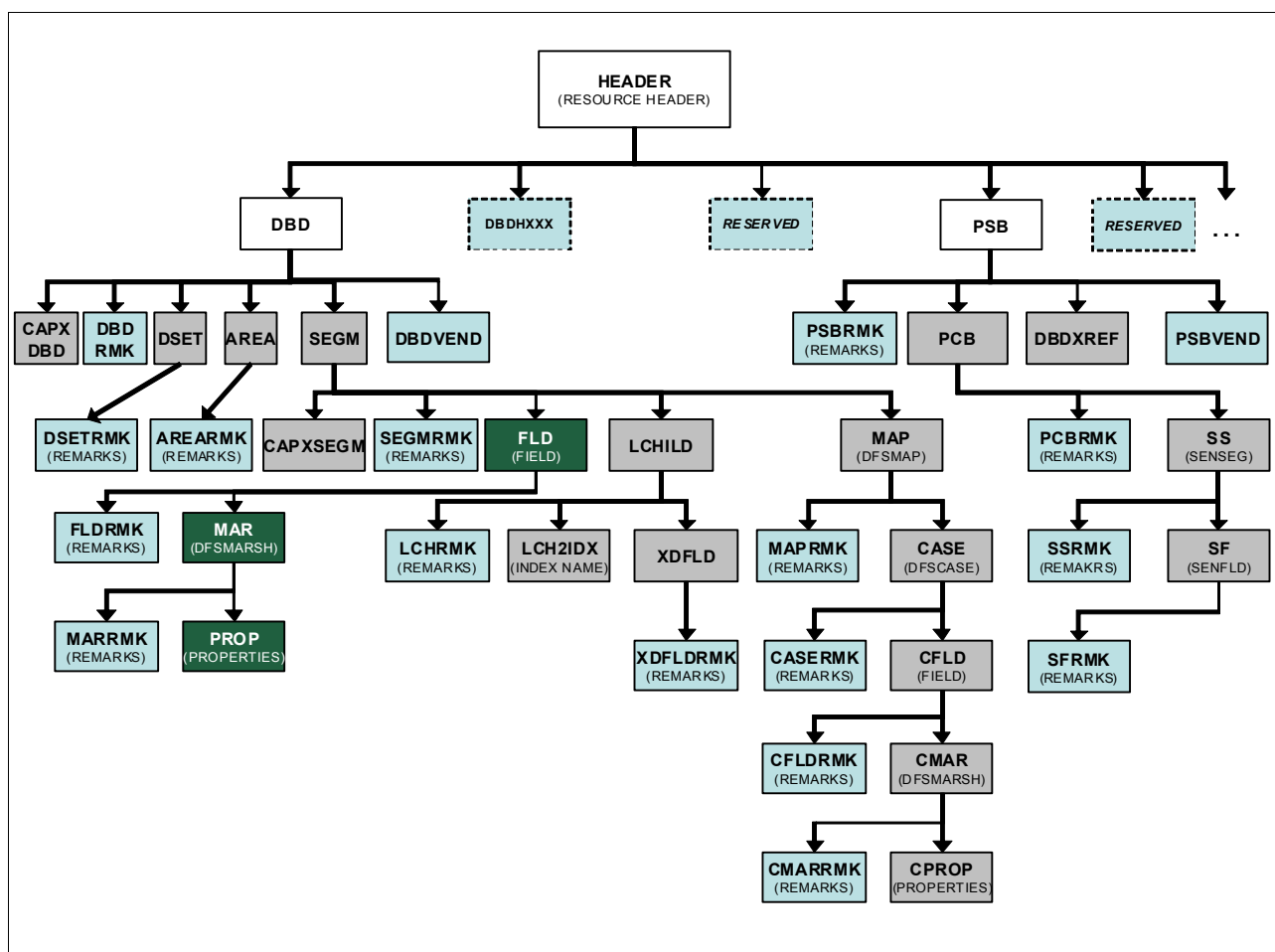


Figure 8-2 Database structure for the catalog database DFSCD000

There are five PSBs that are supplied for accessing and updating the catalog database:

- ▶ DFSCPL00
It is used for an initial load of the database (PROCOPT=L).
- ▶ DFSCP000
It has read-only access (PROCOPT=G) for all segments. This PSB can be used for Assembler and COBOL programs for direct read-only access to the catalog database.
- ▶ DFSCP001
It has update access (PROCOPT=A) for all segments. This PSB can be used for Assembler and COBOL programs for direct update access to the catalog database.

- ▶ DFSCP002

It has read-only access (PROCOPT=G) for all segments. This PSB is used for PL/I programs for direct read-only access to the catalog database.

- ▶ DFSCP003

It has read-only access (PROCOPT=G) for all segments. This PSB is used for PASCAL programs for direct read-only access to the catalog database.

There are several segments in the catalog database, including DBDVEND and PSBVEND, in the database definition for IBM tools and vendor products to store more information. There are also several reserved segments in the database to allow for future expansion.

Most of the segment types (including DBD, DSET, AREA, SEGM, FLD, LCHILD, and XFLD) contain metadata about the specific element that they represent. The data also includes the name of the segment that corresponds to the metadata that is stored.

Other segments (DBDRMK, DSETRMK, AREARMK, SEGMRMK, MAPRMK, and CASERMK) contain remarks about the element definition.

The CAPXDBD and CAPXSEGM segments contain information about Data Capture exit routines that are used.

8.3 IMS catalog database installation and management

The IMS catalog is implemented as a HALDB database. IMS normally requires each HALDB database to be registered in the Database Recovery Control (DBRC). There are companies who choose not to use DBRC in some of their environments. Therefore, a method is provided to define the IMS catalog without the need to register the database in the RECON.

This section describes the following steps for installation and management of the IMS catalog:

- ▶ Installation
- ▶ IMS catalog initial data population
- ▶ ACB generation and changes
- ▶ IMS Catalog Copy utility
- ▶ Keeping multiple versions of metadata in the catalog
- ▶ IMS Catalog Record Purge utility
- ▶ Automatically creating the IMS catalog database data sets
- ▶ Using the IMS catalog without DBRC
- ▶ Aliases and sharing
- ▶ Definitions that are needed for the IMS catalog

IMS provides a number of new utilities to maintain the catalog database when databases or PSB definitions are changed as part of your normal application development lifecycle.

8.3.1 Installation

The first step of the installation of the IMS catalog is to copy the supplied catalog DBD and PSB members from the SDFSRESL data set to your own DBD and PSB libraries. Example 8-2 shows the JCL to perform this copy.

The source for the catalog DBD and PSBs is shipped for reference in the IMS source library SDFSRC. However, the object code in SDFSRESL must be used for execution.

Example 8-2 Copy the supplied DBD and PSB members to your own libraries

```
//CPYMEM EXEC PGM=IEBCOPY
//SDFSRESL DD DSN=IMS12.SDFSRESL,DISP=SHR
//DBDLIB DD DSN=IMS12.IMS12X.DBDLIB,DISP=SHR
//PSBLIB DD DSN=IMS12.IMS12X.PSBLIB,DISP=SHR
//SYSIN DD *
        COPY OUTDD=DBDLIB,INDD=((SDFSRESL,R))
        SELECT MEMBER=(DFSCD000,DFSCX000)
        COPY OUTDD=PSBLIB,INDD=((SDFSRESL,R))
        SELECT MEMBER=(DFSCPL00,DFSCP000,DFSCP001,DFSCP002,DFSCP003)
```

After the DBDs and PSBs are copied to your libraries, you need to build them as ACBs by using the traditional ACB process, as shown in Example 8-3.

Example 8-3 ACBGEN for IMS catalog

```
// JCLLIB ORDER=IMS12.PROCLIB
// EXEC ACBGEN
//SYSIN DD *
        BUILD PSB=(DFSCPL00)
        BUILD PSB=(DFSCP001)
        BUILD PSB=(DFSCP000)
        BUILD PSB=(DFSCP002)
        BUILD PSB=(DFSCP003)
```

Next, create the catalog database data sets. The space that is allocated for the catalog database must be sufficient to store the number of DBD and PSB definitions for your environment. This creation can be done in three ways:

- ▶ The database data sets can be allocated with the *job control language* (JCL) similar to what is shown in Example 8-4.
- ▶ The IMS Catalog Partition Definition Data Set utility (DFS3UCD0) can be used to create the catalog partition definition database data set if DBRC is not in use. It also creates the database data sets for the catalog database.
- ▶ The IMS Catalog Populate utility (DFS3PU00) is used to load or insert records into the catalog database, whether or not DBRC is used for the catalog database. If any of the database data sets do not exist, the utility creates them. The size parameters that are used for automatic creation are specified in the catalog section of the DFSDFxxx procedure library (PROCLIB) member.

Example 8-4 shows the process of defining the database data sets and indexes.

Example 8-4 Define the HALDB data sets, the ILDS, and the primary and secondary indexes

```
//PHIDAM EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
        ALLOCATE DSNAME('IMS12.IMS12X.DFSCDI2D.A00001') FILE(A00001) -
```

```

        RECFM(F,B,S) DSORG(PS) NEW CATALOG -
        SPACE(2,2) CYLINDERS VOLUME(SBOXI5) UNIT(SYSALLDA)

    ALLOCATE DSNAME('IMS12.IMS12X.DFSCDI2D.B00001') FILE(B00001) -
        RECFM(F,B,S) DSORG(PS) NEW CATALOG -
        SPACE(2,2) CYLINDERS VOLUME(SBOXI5) UNIT(SYSALLDA)

    ALLOCATE DSNAME('IMS12.IMS12X.DFSCDI2D.C00001') FILE(C00001) -
        RECFM(F,B,S) DSORG(PS) NEW CATALOG -
        SPACE(2,2) CYLINDERS VOLUME(SBOXI5) UNIT(SYSALLDA)

    ALLOCATE DSNAME('IMS12.IMS12X.DFSCDI2D.D00001') FILE(D00001) -
        RECFM(F,B,S) DSORG(PS) NEW CATALOG -
        SPACE(2,2) CYLINDERS VOLUME(SBOXI5) UNIT(SYSALLDA)

    DEFINE CLUSTER(NAME('IMS12.IMS12X.DFSCDI2D.L00001') -
        FREESPACE(80 10) SHAREOPTIONS(3 3) KEYS(9 0) -
        RECORDSIZE(50 50) SPEED CYLINDERS(1,1) VOLUMES(SBOXI5)) -
        DATA(CONTROLINTERVALSIZE(4096)) -
        INDEX(CONTROLINTERVALSIZE(2048))

    DEFINE CLUSTER(NAME('IMS12.IMS12X.DFSCDI2D.X00001') INDEXED -
        KEYS(16,5) VOL(SBOXI5) REUSE RECORDSIZE (22,22)) -
        DATA(CONTROLINTERVALSIZE(4096))

//PSINDEX EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
    DEFINE CLUSTER (NAME('IMS12Q.IMS12X.DFSCXI2D.A00001') INDEXED -
        SHAREOPTIONS(3 3) KEYS(37,45) REUSE RECORDS(5,5) VOL(SBOXI5) -
        RECORDSIZE(82,82)) DATA(CONTROLINTERVALSIZE(4096))

```

After the database data sets are created, you must update the IMS.PROCLIB(DFSDFxxx) member to define the catalog parameters for your IMS system. There are two sections to be added to the member, depending on which of the following configurations you choose to use:

- ▶ A unique IMS catalog for each system
- ▶ A unique DFSDFxxx member for each system
- ▶ A shared IMS catalog
- ▶ A shared DFSDFxxx member

Example 8-5 shows the simplest environment: A single IMS with a single catalog prefixed with the default name DFSC.

Example 8-5 IMS.PROCLIB(DFSDFxxx) for a single IMS system

```

*-----*
* IMS CATALOG SECTION                                     *
*-----*
<SECTION=CATALOG>
  CATALOG=Y
  ALIAS=DFSC

```

Example 8-6 shows a shared DFSDFxxx member with a separate IMS catalog for each IMS system (defined with an ALIAS that matches the IMSID). The use of aliases for the IMS catalog is described in 8.3.9, “Aliases and sharing” on page 290.

Example 8-6 IMS.PROCLIB(DFSDFxxx) for multiple IMS systems

```
*-----*
* IMS CATALOG SECTION for IMS I12B *
*-----*
<SECTION=CATALOGI12B>
  CATALOG=Y
  ALIAS=I12B
*-----*
* IMS CATALOG SECTION for IMS I12D *
*-----*
<SECTION=CATALOGI12D>
  CATALOG=Y
  ALIAS=I12D
```

The catalog can be activated, by using the new DFSDFxxx member, with a cold start, warm start, or even an emergency restart of the IMS system. The restart is required because IMS reads only the DFSDF member during initialization.

8.3.2 IMS catalog initial data population

After you define the IMS catalog database data sets, you must run a new utility to read the IMS ACBLIB to initially load (populate) the IMS catalog database. As is usual for all new releases of IMS, the ACBLIB must be rebuilt from the DBD and PSB libraries. This rebuild is done by using the type-1 ACB generation utility for the new release before the ACBLIB is used, including populating the catalog. The catalog populate process might need to be done only one time for each system using that IMS catalog.

You start by defining the IMS catalog database either to DBRC, or with the non-DBRC utility. When using DBRC, this process can be done with the batch DBRC utility (DSPURX00) by using the **INIT.DB** and **INIT.PART** commands for the catalog database. The catalog populate process also can be done by using the IMS Partition Definition Utility (PDU) application in the Time Sharing Option (TSO).

DBRC registration is optional. For more information about when DBRC is not used, see 8.3.8, “Using the IMS catalog without DBRC” on page 289.

The initial data population of the IMS catalog is done by using the *IMS Catalog Populate* utility (DFS3PU00), as shown in Figure 8-3.

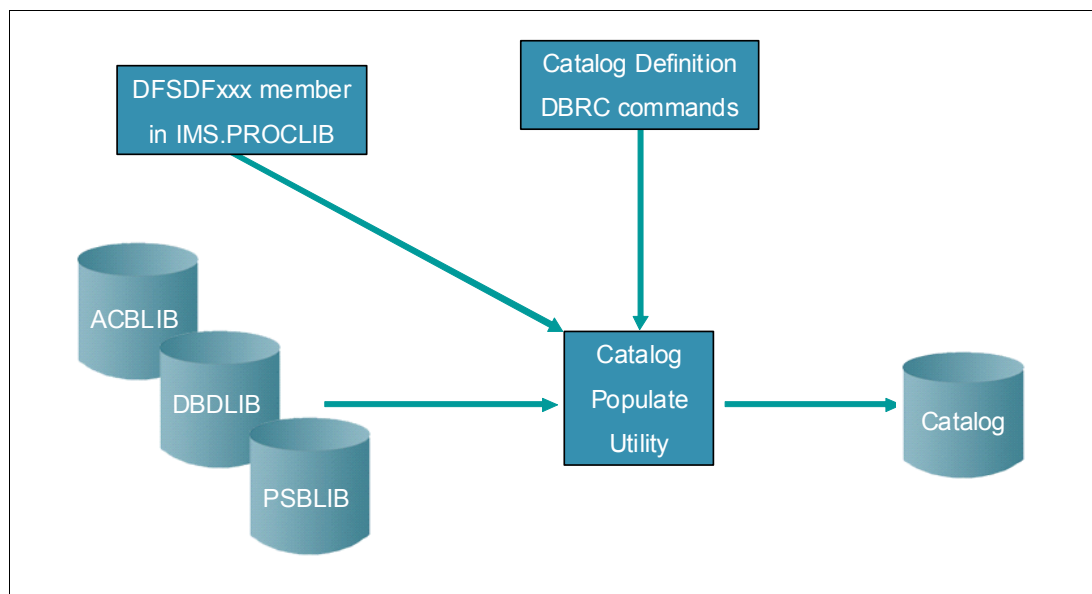


Figure 8-3 *IMS Catalog Populate utility (DFS3PU00)*

The utility reads the IMS.PROCLIB(DFSDFxxx) member, then builds the IMS catalog from the ACB libraries that are referenced in the JCL. The utility also reads the DBD and PSB libraries to capture the information about any generalized sequential access method (GSAM) DBDs. This method is used because they are not built as ACB library members.

After execution, a HALDB REORG record is recorded in the RECON.

The IMS Explorer allows us to add application metadata to these database definitions. This process is needed because at this stage the IMS catalog holds only the rudimentary information that is available from the DBD and PSB definitions. For more information about this process, see 8.5, “The role of the IMS Enterprise Suite Explorer for Development” on page 304.

Example 8-7 shows the JCL that is used to populate an IMS catalog from the current ACB libraries.

Example 8-7 *Running the IMS Catalog Populate utility (DFS3PU00)*

```

//LOADCAT EXEC PGM=DFS3PU00,
// PARM=(DLI,DFS3PU00,DFSCPL00,,,,,,,,,Y,N,,,,,,,,,'DFSDF=12D')
//STEPLIB DD DSN=IMS12.SDFSRESL,DISP=SHR
//DFSRESLB DD DSN=IMS12.SDFSRESL,DISP=SHR
//IMS DD DSN=IMS12.IMS12X.PSBLIB,DISP=SHR
// DD DSN=IMS12.IMS12X.DBDLIB,DISP=SHR
//PROCLIB DD DSN=IMS12.PROCLIB,DISP=SHR
//SYSABEND DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//IEFRDER DD DISP=(,CATLG),DSN=IMS12.IMS12X.PU000.LOG(+1),
// SPACE=(TRK,(5,5),RLSE),UNIT=SYSALLDA
//DFSVSAMP DD DSN=IMS12.PROCLIB(DFSVMDB),DISP=SHR
//IMSACB01 DD DSN=IMS12.IMS12D.ACBLIBA,DISP=SHR
//IMSACB02 DD DSN=IMS12.IMS12D.ACBLIB.DOPT,DISP=SHR
  
```

8.3.3 ACB generation and changes

After you migrate to an IMS system that is using an IMS catalog, you must keep the ACB libraries and the catalog synchronized. IMS uses time stamps to ensure consistency. With time stamps, you can ensure that the catalog data can always be trusted as an accurate equivalent of the metadata that is held in the DBD, PSB, and ACB libraries.

There is a new *ACB Generation and IMS Catalog Populate* utility (DFS3UACB). This utility is used to perform both the generation of ACB members in an IMS.ACBLIB data set and the creation of the corresponding metadata records in the IMS catalog in a single job step. This process is shown in Figure 8-4. The DFS3UACB utility can be used in load mode to initially populate the catalog, or in update mode to add a version of the new or changed definitions. In update mode, a new version of any changed definitions is created in the catalog, rather than altering the existing definitions in the catalog.

When the utility is run in load mode, all existing records in the IMS catalog are discarded.

Figure 8-4 shows the ACB Generation and IMS Catalog Populate utility.

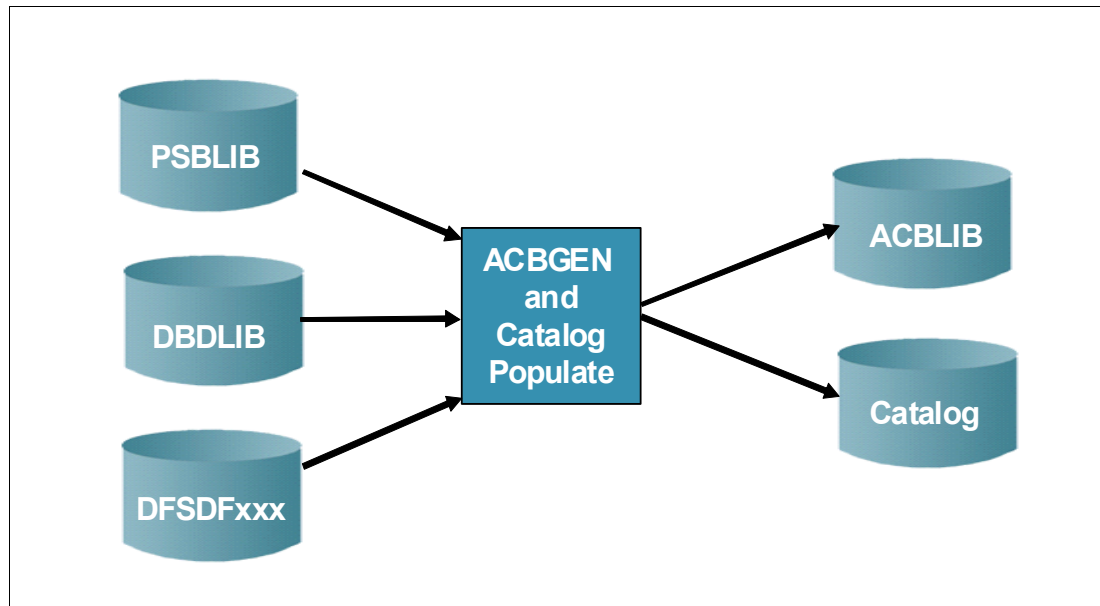


Figure 8-4 ACB Generation and IMS Catalog Populate utility

The new ACB Generation utility writes logs. If the IMS catalog is defined in DBRC, it updates the RECON. It can run as a batch message program (BMP) or DLIBATCH utility. If it is run as a BMP, ensure that the catalog database is opened by IMS for update access. Usually, IMS opens the catalog database in read-only mode.

8.3.4 IMS Catalog Copy utility

After you migrate to an IMS catalog, you must ensure that the IMS catalog is updated every time that you update the IMS ACB library. When you migrate applications from one environment to another, from the test through to production environments, for example, the new *IMS Catalog Copy* utility can be used to keep the metadata synchronized.

This utility allows the export and import of metadata information between catalogs:

- ▶ The utility DFS3CCE0 exports from a catalog and the ACBLIB, DBDLIB, or PSBLIB libraries.

This utility copies an IMS catalog and any included ACB, DBD, and PSB libraries to export data sets in the same job step.

- ▶ The utility DFS3CCI0 imports from the export data set into another catalog, ACBLIB, DBDLIB, and PSBLIB libraries.

At the destination environment, the import module DFS3CCI0 loads or updates an IMS catalog. The module also copies any included ACB, DBD, and PSB libraries from the export data sets into their destination data sets.

For the import function of the IMS Catalog Copy utility, the primary input to the utility is a data set that contains the new copy of the IMS catalog. A CCUCATIM data definition (DD) statement is required to identify this data set.

If the ACB libraries, DBD libraries, and PSB libraries were copied during the export function, they are identified in the import JCL by the following DD statements:

- ▶ **CCUACBIM DD** statement for the ACB library export data set
- ▶ **CCUDBDIM DD** statement for the DBD library export data set
- ▶ **CCUPSBIM DD** statement for the PSB library export data set

The new utilities ensure that the updates to the ACB libraries result in parallel updates to the IMS catalog.

Optionally, the IMS Catalog Copy utility can also create copies of the ACB, DBD, and PSB library members that correspond to the IMS catalog records that are copied.

The IMS Catalog Copy utility creates an import statistics report for the record segments to be loaded or updated in the IMS catalog. When the Catalog Copy utility runs in analysis-only mode, the report reflects only certain statistics. The report reflects only the potential statistics if the IMS catalog were loaded or updated from the ACB libraries that are currently used as input to the utility.

The IMS Catalog Copy utility can be run as either a batch job or as an online BMP.

8.3.5 Keeping multiple versions of metadata in the catalog

The catalog holds multiple versions of the DBD and PSB metadata. The information in the IMS catalog is time stamped. IMS uses these time stamps to keep multiple versions of the metadata.

The catalog section of the DFSDFxxx PROCLIB member specifies the IMS-wide default values for the retention of metadata in the catalog. The parameters define the maximum number of generations and the minimum retention period for which information is to be kept in the catalog. This specification is similar to the way that information is stored by DBRC in the RECON to keep records of a number of Image Copy sets. However, the information is not automatically purged, as it is with DBRC. The DBDs and PSBs are only deleted by the *IMS Catalog Record Purge* utility (DFS3PU10). The retention parameters for specific databases can be specified with DFS3PU10.

The following two parameters are used to control the optional **RETENTION** statement in the DFSDFxxx catalog section: VERSIONS=nnn and DAYS=ddd:

- ▶ **VERSIONS** defines the maximum number of versions of a DBD or PSB to be kept in the IMS catalog database. The value can be 1 - 65535; the default value is 2. When the maximum value is reached, the oldest version (based on the ACBGEN time stamp) is replaced by the newest version.
- ▶ **DAYS** defines the minimum number of days a version remains in the catalog. The value can be 0 - 65535; the default value is 0 (function disabled). When a version of the catalog metadata is older than the specified version, it becomes a candidate for removal. It might be removed when new versions of the same DBD or PSB are added to the IMS catalog.

These parameters work in the same way that GENMAX and RECOVPD work with DBRC: If the maximum number of versions is reached but the retention period has not expired, a new catalog record is kept. The oldest record is not removed.

Example 8-8 shows the specification in the IMS.PROCLIB(DFSDFxxx) member, where at least five generations of metadata are kept, for a minimum of 30 days.

Example 8-8 Setting default maximum generations and retention periods for catalog metadata

```
<SECTION=CATALOG>
CATALOG=Y
ALIAS=DFSC
RETENTION=(VERSIONS=5,DAYS=30)
```

When multiple versions are stored in the catalog, if an online change is reversed, the catalog might still have the previous version available.

For logically related databases, where in the past the DBD information is kept only in the DBDLIB and not the ACBLIB, the Catalog Populate utility sets the time stamp to zero. The time stamp is set to zero until the combined ACBGEN/Populate utility runs for those databases.

Metadata information in the catalog can be removed by the IMS Catalog Record Purge utility (DFS3PU10) if it is outdated or if the version exceeds the specified retention value. When using DFS3PU10, the default of two versions applies. If no retention value is specified, each execution of ACBGEN or Populate Utility adds a version.

8.3.6 IMS Catalog Record Purge utility

The IMS Catalog Record Purge utility (DFS3PU10) provides several functions:

- ▶ Sets the record retention criteria for specific DBD and PSB records in the catalog database.
- ▶ Produces a list of DBDs and PSBs with versions that are no longer needed according to the current retention criteria for each record.
- ▶ Purges specific versions (based on the ACB time stamp) of a record for a DBD or PSB resource from the IMS catalog database.
- ▶ Purges unnecessary versions of IMS catalog records from the catalog database based on criteria that you specify.

8.3.7 Automatically creating the IMS catalog database data sets

Use the DFSDFxxx member of the IMS PROCLIB data set to specify processing options for the IMS catalog. The DFSDFxxx member contains parameters that are organized into sections. Example 8-9 shows the specify options for the IMS catalog. In a data sharing environment, this section defines the IMS catalog for all IMS systems that are not individually configured with a CATALOGxxxx section.

Example 8-9 IMS catalog parameters in DFSDFxxx

```
<SECTION=CATALOG>
CATALOG=Y
ALIAS=DFSC
/* IXVOLSER=vvvvvv if DFSMS is not used */
DATACLAS=IMSDATA
MGMTCLAS=EXTRABAK
STORCLAS=IMS
SPACEALLOC=500
```

The additional parameters allow you to define the DFSMS DATALCLAS, MGMTCLAS, STORCLAS, or volume serial number if DFSMS is not used.

SPACEALLOC=nnnn (0 - 9999; the default is 500) also allows you to define the space as a percentage of the predefined IMS value.

8.3.8 Using the IMS catalog without DBRC

The IMS catalog database is a standard PHIDAM database. Usually, that type of database requires registration to DBRC. DBRC tracks the logs that are used when the database is updated. DBRC can also be used to generate jobs to perform image copies and accumulate changes with the IMS Change Accumulation utility.

You might be running your IMS system with non-HALDB databases that are not registered to DBRC (which is a common configuration for test systems). If the databases are not registered, IMS does not insist that the catalog HALDB database (and index) is registered in RECON. If the HALDB is not registered, IMS does not track logs, change accumulations, or image copies.

The backup, logging, and recovery of the IMS catalog is the responsibility of the user. This process can be done by running stand-alone IBM MVS utilities or by DFSMSHsm backup and recovery. In some cases, the usage of the IMS Catalog Populate utility to rebuild the database might be the best choice.

To overcome the need for IMS to have the IMS catalog PHIDAM database metadata (normally stored in the DBRC RECON), IMS has special handling for the IMS catalog. Instead of having database, partition, and database data set records that are defined in the RECON, IMS uses dynamic allocation, the DFSMDA macro, with the **TYPE=CATDBDEF** statement. This statement defines the dynamic allocation parameter list for the IMS catalog partition definition data set. This data set contains the definitions for the catalog HALDBs that are not defined in the DBRC RECON data set. The DD name of the catalog partition definition data set is DFSHDBSC.

A new utility, DFS3UCD0, is used to create the HALDB partition definition data set. You need to supply a system input stream (SYSIN) data set to define the data that normally is used on the **INIT.DB** and **INIT.PART** statements when a database is registered in DBRC.

Example 8-10 shows an example. The structure information is stored in the definition data set (DFSHDBSC).

Example 8-10 Example SYSIN for DFS3UCD0

```
HALDB=(NAME=DFSCD000,HIKEY=YES)

PART=(NAME=DFSCD000,
      PART=DFSCD01,
      DSNPREFIX=IMSTESTS.DFSCD000,
      KEYSTHEX=FFFFFFFFFFFFFFFFFFFFFFFFFFFF)

HALDB=(NAME=DFSCX000,HIKEY=YES)

PART=(NAME=DFSCX000,
      PART=DFSCX01,
      DSNPREFIX(IMSTESTS.DFSCX000)
      KEYSTHEX=FFFFFFFFFFFFFFFFFFFFFFFFFFFF)
```

The next thing that you need to add is a statement in the IMS.PROCLIB(DFSDFxxx) member to define the IMS catalog that is not registered in DBRC (Example 8-11).

Example 8-11 IMS.PROCLIB(DFSDFxxx) definition for a non-DBRC IMS catalog

```
<SECTION=DATABASE>
UNREGCATLG=DFSCD000
```

Optionally, to avoid making JCL changes, you can also build a DFSMDA dynamic allocation macro to define the DFSHDBSC to IMS, as shown in Example 8-12. **TYPE=CATDBDEF** is a new specification just for an unregistered IMS catalog database.

Example 8-12 DFSMDA definition

```
DFSMDA TYPE=INITIAL
DFSMDA TYPE=CATDBDEF,DSNAME=IMS12.IMS12D.CATALOG.DEFDS
DFSMDA TYPE=FINAL
END
```

8.3.9 Aliases and sharing

In an IMSplex, you have a number of options for how you set up the ACBLIB and the IMS catalog. You can share or clone the ACB libraries ACBLIBA or ACBLIBB. Similarly, the IMS catalog can be shared, or it can use a separate IMS catalog for each IMS system.

The way that ACB libraries in IMS V10 or IMS V11 are used depends on whether you are using local, global, and member online changes. IMS V12 and later supports these configuration options, and you can configure the IMS catalog to participate with your existing shared or discrete ACB libraries. The IMS catalog can be shared between multiple IMS systems.

Tip: Convert your systems to use global online change, which dynamically allocates ACB libraries and the usage of member online changes.

Figure 8-5 shows the migration of a system from an IMSplex where each IMS has its own cloned ACB library pair. However, we also introduce a single shared IMS catalog that is populated from all the existing ACB libraries.

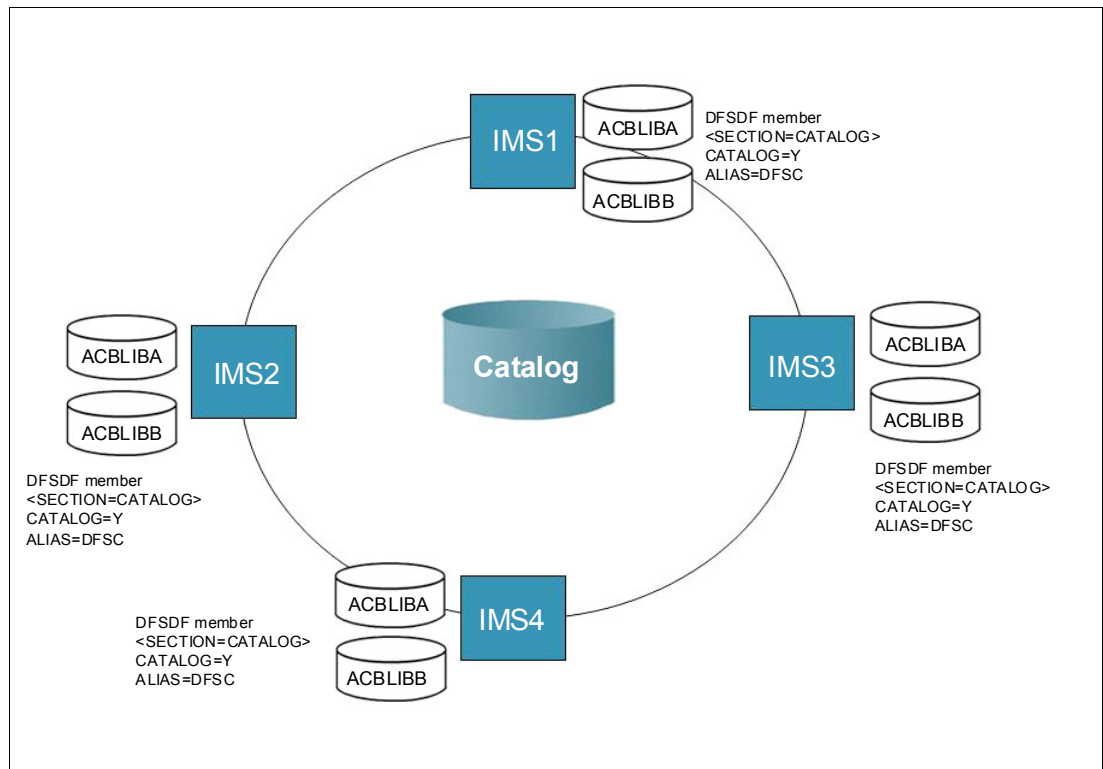


Figure 8-5 Multiple IMS, cloned ACBLIB, and shared IMS catalog

Figure 8-6 shows a system where you start from separate (cloned) ACB libraries. In this example, we chose to create a unique aliased IMS catalog for each IMS system.

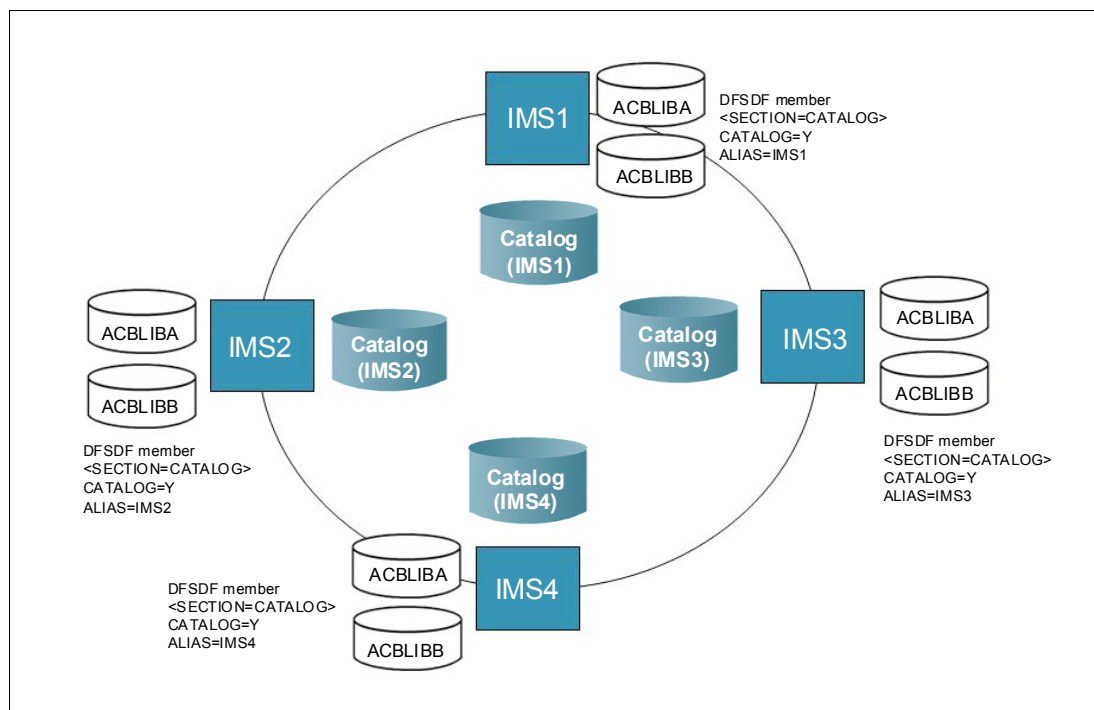


Figure 8-6 Multiple IMS, cloned ACBLIB, and one IMS catalog for each IMS

Figure 8-7 shows the integrated system, where there is one pair of ACB libraries for the IMSplex and a single shared IMS catalog (registered in DBRC with SHARELVL=3). In this system, global online change and member online change are used.

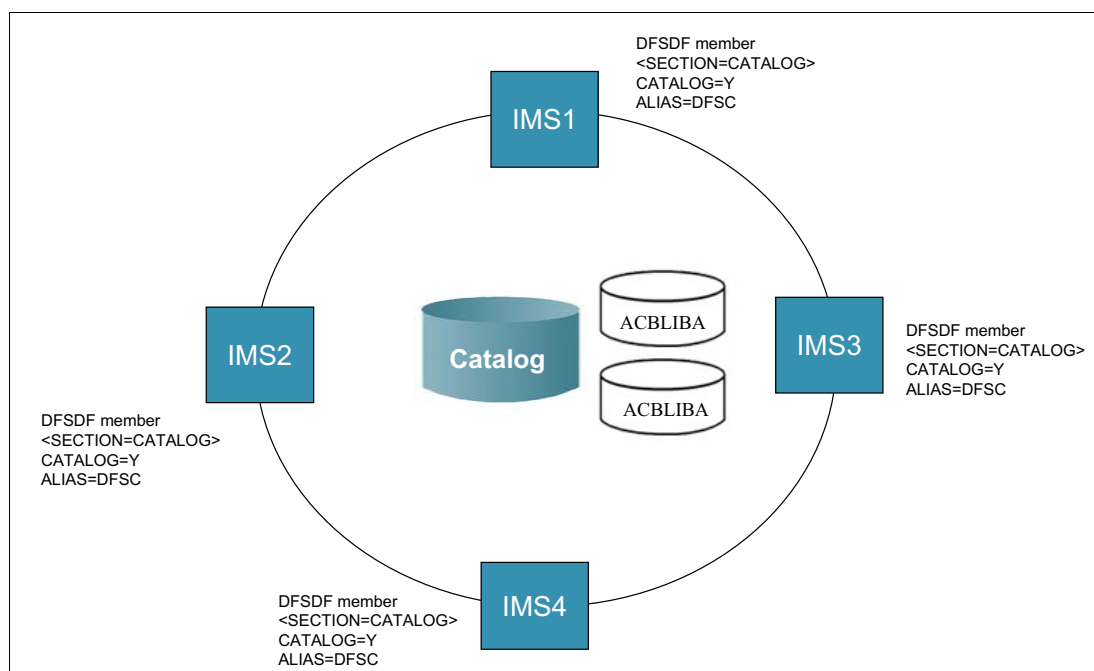


Figure 8-7 Multiple IMS, shared ACBLIBs, and shared IMS catalog

IMS handles the loading of the DBDs (DFSCD000 and DFSCX000) and the PSBs (DFSCP000, DFSCP001, DFSCP002, and DFSCP003). IMS also makes the internal changes that are needed when aliased names are used.

8.3.10 Definitions that are needed for the IMS catalog

Before you can create the IMS catalog and initially load it with the populate utility, you must update the IMS system to use it by adding statements to the IMS.PROCLIB(DFSDFxxx) member.

You have some flexibility in how to define the CATALOG section in the DFSDFxxx member. A simple single IMS is shown in Example 8-13 with its suggested ALIAS name.

Example 8-13 Single IMS system

```
<SECTION=CATALOG>
CATALOG=Y
ALIAS=DFSC
```

Alternatively, if you have two or more IMS systems that use shared queues, you might want them to share a single IMS catalog and the same DFSDFxxx proclib member. In this case, you can define different section suffixes in the **CATALOG** statement by using the IMS ID, and you can define the catalog database with a single shared ALIAS name, as shown in Example 8-14.

Example 8-14 Multiple IMS systems

```
<SECTION=CATALOGI12A>
CATALOG=Y
ALIAS=I12Q
<SECTION=CATALOGI12C>
CATALOG=Y
ALIAS=I12Q
```

8.4 Application usage of the catalog

IMS V12 and later can access the IMS catalog when IMS metadata is required. IMS also provides a user interface to the IMS catalog data and a new DL/I call to access the IMS catalog. One source of metadata for populating the catalog is the DBDs and PSBs. The extensions of the metadata information and the additional data manipulation possibilities that are introduced require more information than was possible with the DBD and PSB sources before IMS V12. As a result, several DBDGEN and PSBGEN macros were enhanced in IMS V12.

Details about using the IMS catalog are provided in the following sections:

- ▶ DBD and PSB source changes
- ▶ Get Unique Record DL/I call
- ▶ IMS catalog access
- ▶ SSA enhancements

8.4.1 DBD and PSB source changes

There are new **DBDGEN** statements in IMS V12 and changes to existing ones. The new and changed statements extend the metadata information available to the catalog. Much of this metadata information is the basis for the extended Object exploitation by the Java language.

DFSMARSH statement in the DBD

The **DFSMARSH** statement in a DBD defines the marshalling attributes for field data. The **DFSMARSH** statement must immediately follow the **FIELD** statement to which it applies. You can use the **DFSMARSH** statement to specify the following data marshalling attributes for the data that is contained in a field:

- ▶ You can specify the code page or character encoding that defines the character data in a field.
- ▶ You can specify a data-conversion routine for IMS to use when converting field data. For example, use it when converting field data from the data type that IMS uses to physically store data, to a data type that is expected by an application program.
- ▶ You can specify whether a numeric data type is signed or not with the **ISSIGNED** parameter.
- ▶ The pattern to use for dates and times can be specified with the **PATTERN** parameter.
- ▶ You can specify the properties that are used with a user-provided data-conversion routine.

The MAR segment type in the catalog database contains information about a field marshaller definition in an IMS database. Each FLD segment can have a MAR child segment that contains data marshalling properties for that field. The following information in this segment type is generated from the input parameters of the **DFSMARSH** statement of the DBDGEN utility:

- ▶ **ENCODING**

Specifies the default encoding of all character data in the database that is defined by this DBD.

Default **ENCODING** is CP1047 (EBCDIC). This default value can be overridden for individual segments and fields.

- ▶ **USERTYPECONVERTER**

Specifies the fully qualified Java class name of the user-provided Java class to be used for type conversion.

- ▶ **INTERNALTYPECONVERTER**

Specifies the internal conversion routine that IMS uses to convert the IMS data into Java objects for Java application programs. IMS requires the specification of either **INTERNALTYPECONVERTER** or **USERTYPECONVERTER**, but not both. Valid values for the **INTERNALTYPECONVERTER** parameter include:

ARRAY, BINARY, BIT, BLOB, BYTE, CHAR, CLOB, DOUBLE, FLOAT, INT, LONG, PACKEDDECIMAL, SHORT, STRUCT, UBYTE, USHORT, UINT, ULONG, XML_CLOB, and ZONEDDECIMAL.

- ▶ **ISSIGNED**

Valid only for DATATYPE=DECIMAL. Valid values are Y (default) or N.

- ▶ **PROPERTIES**

Specifies properties for user type converter names with the **USERTYPECONVERTER** parameter. These properties are passed to the user type converter.

► **PATTERN**

An optional field that specifies the pattern to use for the date, time, and time stamp Java data types.

The **PATTERN** parameter applies only when the **DATE**, **TIME**, or **TIMESTAMP** is specified on the **DATATYPE** keyword in the **FIELD** statement. Also, **CHAR** must be specified on the **INTERNALTYPECONVERTER** keyword in the **DFSMARSH** statement.

DFSMAP statement

The **DFSMAP** statement enables the alternative mapping of fields within a segment.

The **DFSMAP** statement defines a group of one or more map cases and relates the cases to a control field. The control field identifies which map case is used in a particular segment instance, as shown in Figure 8-8.

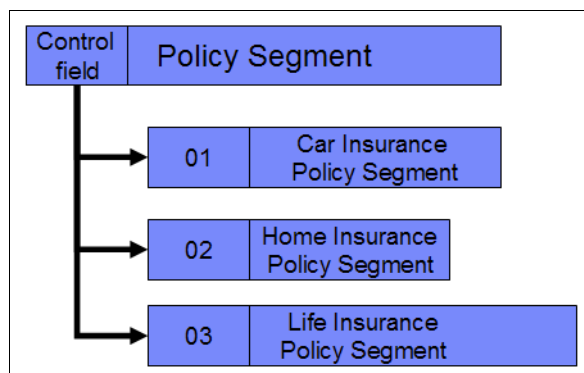


Figure 8-8 Several mappings for the same segment

► **NAME**

Defines the name of this map.

► **DEPENDING ON**

External name of the control field within this segment that contains the value that determines which map case is used for a particular segment instance. If the control field does not contain a value that corresponds to a **CASEID** in a **DFSCASE** statement for this map, the map is not used for this segment instance.

If the **FIELD** statement that defines the control field does not explicitly code the **EXTERNALNAME** parameter, specify the value of the **NAME** parameter in the **DEPENDING ON** field.

► **REMARKS**

Allow for a comment up to 256 characters long to be recorded in the DBD and the IMS catalog database.

DFSCASE statement

Many applications allow for several different layouts for the same segment type in an IMS database. Typically, this configuration is done in COBOL with the **REDEFINES** statement, and in PL/I with the **DEFINED** statement. The corresponding facility in IMS is the **DFSCASE** statement, which defines a map case, that is, a conditional mapping of a set of fields in a segment. Each map case has a name and an identifier (ID). A single segment can be defined with multiple map cases. The map cases within a segment are grouped by the **DFSMAP** statement, which also associates the map cases with their control field.

Each map case is defined with a case ID. The case ID is stored in the control field to identify which case map is used for a particular segment instance. Typically, the control field is defined at the beginning of the segment before the **DFSMAP** statement.

The fields that make up a map case identify the map case that they belong to by the name on the **CASENAME** parameter in the **FIELD** statement. **CASENAME** is valid and required only to associate a **FIELD** statement with the preceding **DFSCASE** statement that defines the map case to which this field belongs. The value of **CASENAME** must match the value that is specified on the **NAME** parameter of the **DFSCASE** statement.

The ID of a map case is specified on the **CASEID** parameter. In a segment instance, the ID is inserted in a control field to indicate which map case is in use:

- ▶ **NAME**
Defines the name of this case.
- ▶ **CASEIDTYPE**
Defines the data type of the value that is specified in the **CASEID** parameter.
- ▶ **MAPNAME**
The name of the map that this case belongs to, as specified on the **NAME** parameter in the **DFSMAP** statement.
- ▶ **CASEID**
Defines a unique identifier for the case. A segment instance specifies the **CASEID** value in a user-defined control field when part or all of the field structure of the segment is mapped by this case. The **CASEID** must be unique within the map to which the case belongs.
- ▶ **REMARKS**
Allow for a comment of up 256 characters to be recorded in the DBD and the IMS catalog database.

Enhancements to DBD definition statements

There are a number of enhancements to the statements that are used to describe a database definition with IMS V12.

Changes to the DBD statement

ENCODING specifies the default encoding of all character data in the database that is defined by this DBD; otherwise, the default **ENCODING** is CP1047 (EBCDIC). The value can be overridden in individual segments or fields.

Changes to the SEGM statement

- ▶ **ENCODING**
Specifies the default encoding of all character data in the database that is defined by this DBD; otherwise, the default **ENCODING** is CP1047 (EBCDIC). The value can be overridden in individual fields.
- ▶ **EXTERNALNAME**
This parameter is used to give a segment an extended (1 - 128 byte) name.

Changes to the FIELD statement

- ▶ **NAME**
Specifies the name of this field within a segment type. The name must be provided to be able to search on the field.

For key-sequenced field types and field types that are referenced by an **XDFLD** statement, the **NAME** parameter is required.

For other field types, you can optionally omit the **NAME** parameter when the **EXTERNALNAME** parameter is specified.

► **PARENT**

Specifies the name of a field that is defined as a structure or array in which this field is contained. Must be the value of the **EXTERNALNAME** parameter in the definition of the referenced field.

The referenced field must be defined with either **DATATYPE=ARRAY** or **DATATYPE=STRUCT**.

► **CASENAME**

The name of the map case to which this field belongs.

This parameter is required only to associate a field statement with the preceding **DFSCASE** statement that defines the map case.

This parameter must match the value that is specified on the **NAME** parameter of the **DFSCASE** statement.

► **DATATYPE**

This parameter is an optional value that specifies one of the following external data types for the field:

ARRAY, BINARY, BIT, BYTE, CHAR, DATE, DECIMAL (with precision and scale), DOUBLE, FLOAT, INT, LONG, OTHER, SHORT, STRUCT, TIME, TIMESTAMP, and XML.

► **REDEFINES**

This is a parameter that specifies the name of a field in the segment. The field must be the same length as the field that is redefined.

The field cannot be an ARRAY or contain an ARRAY.

► **EXTERNALNAME**

This parameter is an optional alias for the **NAME=** parameter. Java application programs use the external name to refer to the field.

The **EXTERNALNAME** parameter is required only when the **NAME** parameter is not specified. If the **NAME** parameter is not specified, you cannot search on this field.

► **DEPENDSON**

This parameter specifies the name of a field that defines the number of elements in a dynamic array. Referenced fields must precede the field statement that specifies this parameter.

The name that is specified must be the value, whether explicitly defined or accepted by default, of the **EXTERNALNAME** parameter in the definition of the referenced field.

The **DEPENDSON** parameter is valid only when **DATATYPE=ARRAY** is also specified.

The field that is referenced by the **DEPENDSON** parameter must be defined with one of the following **DATATYPE** values: INT, SHORT, LONG, DECIMAL.

► **MINOCCURS**

Valid for **DATATYPE=ARRAY** only, **MINOCCURS** is a required numeric value that specifies the minimum number of elements in an ARRAY. **MINOCCURS** is invalid for all other data types.

► **MAXOCCURS**

Valid for **DATATYPE=ARRAY** only, **MAXOCCURS** is a required numeric value that specifies the maximum number of elements in an ARRAY.

MAXOCCURS must be greater than or equal to **MINOCCURS**, but not zero.

► **MAXBYTES**

This parameter specifies the maximum size of a field in bytes when the byte-length of the field instance can vary based on the number of elements in a dynamic array. **MAXBYTES** and **BYTES** are mutually exclusive.

The value of **MAXBYTES** must be greater than or equal to the maximum total of the byte values of all fields nested under this field.

The **MAXBYTES** parameter is required and valid only in the following cases:

- The field is defined as a dynamic array.
- For a field defined as a static array or for a structure that contains a nested field that is defined as a dynamic array.

► **STARTAFTER**

This parameter specifies the name of the field that directly precedes this field in the segment. The name that is specified must be the value of the **EXTERNALNAME** parameter in the definition of the referenced field.

STARTAFTER is required and valid only when the starting position of a field cannot be calculated. The starting position cannot be calculated because the field is preceded at a prior offset by a field that is defined as a dynamic array.

The **STARTAFTER** parameter cannot be specified on fields that define an array field as a parent.

► **RELSTART**

This parameter specifies the starting position of a field that is defined as an element of an array or a structure:

- For fields that specify an array field as a parent, **RELSTART** is required.
- For fields that specify a structure as a parent, **RELSTART** is required only when a dynamic array precedes the structure at any prior offset in the segment.

RELSTART is the starting byte offset of the field relative to the start of the array or structure.

Changes to remarks for DBDs and PSBs

The following existing IMS macros are updated to allow up to 256 characters to be stored as a remark for a DBD or PSB. This update allows the user to specify comments that are stored in the DBDLIB, PSBLIB, ACBLIB, and IMS catalogs. This notation helps ensure that comments are not lost if the source is lost:

► PSB remarks

PCB, SENFLD, and SENSEG

► DBD remarks

DBD, SEGM, FIELD, XDFLD, LCHILD, DATASET, and AREA

8.4.2 Get Unique Record DL/I call

IMS V12 provides a new DL/I call specifically for the IMS catalog. The *Get Unique Record* (GUR) call retrieves the metadata for an IMS DBD or PSB from the catalog database. To read all the segments in this database record, the GUR call functions similarly to a Get Unique (GU) call followed by a series of Get Next within Parent (GNP) calls. This call can be used only to retrieve the metadata definition of an IMS DBD or PSB from the catalog database.

The GUR call reads an entire database record from the catalog database and returns an XML document that contains the metadata definition for the requested IMS resource (DBD or PSB). If the XML document is larger than the IOAREA provided by the application, subsequent GUR calls can be issued to retrieve the balance of the XML document. This process is done by passing back a token that was returned by the initial GUR call. The data is buffered by IMS. This buffering is based on the assumption that the entire XML document is needed to minimize the processor usage when you ask for subsequent blocks of data to be returned. The advantage is that you do not need to overestimate the size of the IOAREA. If the IOAREA is too small, the call still succeeds and the balance of the data can be requested without having to reread the catalog database. This new call simplifies access to the database definition metadata and minimizes the processor usage when retrieving it.

If you pass a token with a value of zero, IMS assumes that it is a new GUR call and starts from the beginning.

DFSDDLTO and IMS Restructured Extended Executor (REXX) were updated to add special processing for the IMS catalog and the XML that is returned from a GUR call.

The GUR call builds an entire XML instance document from the information in the catalog database. Internally, each GUR call with a zero AIBRTKN runs a GU and multiple GNP calls to read the IMS catalog. When the I/O area is too small for output, the output buffer is kept. A token is returned in the application interface block (AIB) to allow the application to resume copying data from the buffer to I/O area on subsequent calls.

GUR reads an entire record. SSAs can be coded starting with the HEADER and then the DBD or PSB. GUR does not function like GU calls in which you can read a segment with one qualified SSA, then get a lower level segment with a different SSA.

The GUR call can be used only to access an IMS catalog database, and the call requires the AIB interface. If an attempt is made to use the GUR call through the ASMTDLI, CBLTDLI, or PLITDLI interfaces, IMS returns a "DE" status code.

During initialization, IMS reads the time stamps from the ACBLIB and stores them into the database control blocks. The GUR call uses the time stamp to find the active member that is used by IMS, or the call finds the first record only, when IMS does not have a time stamp. IMS does not have a time stamp for resources that are not defined to this IMS.

Segment search arguments that are used with the GUR call

Typically, segment search arguments (SSAs) are needed when using the GUR call to identify the database or PSB definition to be retrieved. SSAs are specified on the GUR call in the same way as other GU calls to IMS, that is, **field <relational operator> field-value**.

Example 8-15 shows an application program that is using the GUR call to retrieve the metadata for the S2U1DBD database.

Example 8-15 Sample application that uses the GUR DL/I call

```

Identification division.
    program-id. gur.
Environment division.
Data division.
    Working-storage section.
    01 dli-insert pic x(4) value 'ISRT'.
    01 dli-gur    pic x(4) value 'GUR'.
    01 dli-gu     pic x(4) value 'GU'.
    01 header-ssa.
        02 filler          pic x(8) value 'HEADER  '.
```

```

02 filler          pic x(1)  value '('.
02 ssa-field-name  pic x(8)  value 'RHDRSEQ '.
02 ssa-boolean     pic x(2)  value '= '.
02 ssa-field-value pic x(16) value 'DBD      S2U1DBD '.
02 filler          pic x(1)  value ')'.
01 AIB.
02 AIBRID          PIC x(8).
02 AIBRLen        PIC 9(9)  USAGE BINARY.
02 AIBRSFUNC       PIC x(8).
02 AIBRSNM1        PIC x(8).
02 AIBRSNM2        PIC x(8).
02 AIBRESV1        PIC x(8).
02 AIBOALEN        PIC 9(9)  USAGE BINARY.
02 AIBOAUSE        PIC 9(9)  USAGE BINARY.
02 AIBRESV2        PIC x(12).
02 AIBRETRN        PIC 9(9)  USAGE BINARY.
02 AIBREASN        PIC 9(9)  USAGE BINARY.
02 AIBERRXT        PIC 9(9)  USAGE BINARY.
02 AIBRESA1        USAGE POINTER.
02 AIBRESA2        USAGE POINTER.
02 AIBRESA3        USAGE POINTER.
02 AIBRESV4        PIC x(40).
02 AIBRSAVE        OCCURS 18 TIMES USAGE POINTER.
02 AIBRTOKN        OCCURS 6 TIMES  USAGE POINTER.
02 AIBRTOKC        PIC x(16).
02 AIBRTOKV        PIC x(16).
02 AIBRTOKA        OCCURS 2 TIMES PIC 9(9)  USAGE BINARY.
01 gur-returned-data pic x(32000).
Linkage section.
1 io-pcb.
3 filler          pic x(60).
1 db-pcb.
3 filler          pic x(10).
3 status-code     pic x(2).
3 filler          pic x(20).
Procedure division using io-pcb db-pcb.
move "DFSAIB "    to AIBRID.
move length of aib to AIBRLen.
move length of gur-returned-data to AIBOALEN.
move spaces       to AIBRSFUNC.
MOVE "DFSCAT00"   TO AIBRSNM1.
call 'AIBTDLI' using dli-gur aib
           gur-returned-data header-ssa.
set address of db-pcb to aibresal.
display '|' status-code '|' db-pcb.
display 'aib return is ' AIBRETRN .
display 'aib reason is ' AIBREASN .
display gur-returned-data.
goback .
End program gur.

```

The first few lines of the data that is returned by the GUR call from the program (shown in Example 8-15 on page 299), are shown in Example 8-16 on page 301. The first 56 characters in the IOAREA are not part of the XML document.

Example 8-16 Sample data that is returned by a GUR call

```
%      ?>      > ? >      /> /%?>      `      <ns2:dbd xmlns:ns2="http://www.ibm.com/ims/DBD" dbdName="S2U1DBD"
" timestamp="1215015125765" version="02/10/1114.51" xmlSchemaVersion="1"><access dbType="HIDAM"><hidam datxexit="N" pass
word="N" osAccess="VSAM"><dataSetContainer><dataSet ddname="S2U1DB" label="DSG1" searchA="0" scan="3"><block size="0"/>
<size size="0"/><frspc fspf="0" fbff="0"/></dataSet></dataSetContainer></hidam></access><segment imsName="CUSTROOT" name
="CUSTROOT" encoding="Cp1047"><hidam label="DSG1"><bytes maxBytes="76"/><rules insertionRule="L" deletionRule="L" replac
ementRule="L" insertionLocation="LAST"/><pointer physicalPointer="TWINBWD" lparnt="N" ctr="N" paired="N"/></hidam><field
imsDatatype="C" imsName="CUSTNO" name="CUSTOMERNUMBER" seqType="U"><startPos>1</startPos><bytes>4</bytes><marshaller><t
ypeConverter>BINARY</typeConverter></marshaller><applicationDatatype datatype="BINARY"/></field><field imsDatatype="C" n
ame="FIRSTNAME"><startPos>5</startPos><bytes>10</bytes><marshaller encoding="Cp1047"><typeConverter>CHAR</typeConverter>
</marshaller><applicationDatatype datatype="CHAR"/></field><field imsDatatype="C" name="LASTNAME"><startPos>15</startPos
><bytes>20</bytes><marshaller encoding="Cp1047"><typeConverter>CHAR</typeConverter></marshaller><applicationDatatype dat
atype="CHAR"/></field><field imsDatatype="C" name="DATEOFBIRTH"><startPos>35</startPos><bytes>10</bytes><marshaller enco
ding="Cp1047"><typeConverter>CHAR</typeConverter></marshaller><applicationDatatype datatype="CHAR"/></field><field imsDa
tatype="C" name="HOUSENAME"><startPos>45</startPos><bytes>20</bytes><marshaller encoding="Cp1047"><typeConverter>CHAR</t
ypeConverter></marshaller><applicationDatatype datatype="CHAR"/></field><field imsDatatype="C" name="HOUSENUMBER"><start
```

A better way to display this XML information is through a browser. Example 8-17 shows the same information as displayed by Microsoft Internet Explorer.

Example 8-17 Sample data that is returned by a browser

```
<ns2:dbd xmlns:ns2="http://www.ibm.com/ims/DBD" dbdName="S2U1DBD" timestamp="1215015125765"
version="02/10/1114.51" xmlSchemaVersion="1">
  <access dbType="HIDAM">
    <hidam datxexit="N" password="N" osAccess="VSAM">
      <dataSetContainer>
        <dataSet ddname="S2U1DB" label="DSG1" searchA="0" scan="3">
          <block size="0" />
          <size size="0" />
          <frspc fspf="0" fbff="0" />
        </dataSet>
      </dataSetContainer>
    </hidam>
  </access>
  <segment imsName="CUSTROOT" name="CUSTROOT" encoding="Cp1047">
    <hidam label="DSG1">
      <bytes maxBytes="76" />
      <rules insertionRule="L" deletionRule="L" replacementRule="L" insertionLocation="LAST" />
      <pointer physicalPointer="TWINBWD" lparnt="N" ctr="N" paired="N" />
    </hidam>
    <field imsDatatype="C" imsName="CUSTNO" name="CUSTOMERNUMBER" seqType="U">
      <startPos>1</startPos>
      <bytes>4</bytes>
      <marshaller>
        <typeConverter>BINARY</typeConverter>
      </marshaller>
      <applicationDatatype datatype="BINARY" />
    </field>
    <field imsDatatype="C" name="FIRSTNAME">
```

```

<startPos>5</startPos>
<bytes>10</bytes>
<marshaller encoding="Cp1047">
<typeConverter>CHAR</typeConverter>
</marshaller>
<applicationDatatype datatype="CHAR" />
</field>

```

The XML schemas for the documents that are returned as responses to this call are included in the IMS.ADFSSMPL data sets:

- ▶ DFS3XDBD.xsd (for DBD records)
- ▶ DFS3XPSB.xsd (for PSB records)
- ▶ SSAs can be specified only up to the DBD or PSB level in the catalog database

No more than two SSAs can be specified:

- ▶ One for the root HEADER segment
- ▶ One for the DBD or PSB segment

Example 8-18 shows the IMS Test Program (DFSDDL0) statements that are needed to process a GUR call. The **DATA** statement is needed to provide sufficient IO-AREA for the XML document that contains the database definition to be returned to DFSDDL0.

Example 8-18 DFSDDL0 statements for GUR

```

S 1 1 1 1 1 DFSCD000 AIB
L GUR HEADER (RHDRSEQ ==DBD S2U1DBD )
L Z9999 DATA

```

8.4.3 IMS catalog access

When an IMS application program requires access to the metadata in the catalog, a PSB to access the catalog database is automatically attached to the PSB that is loaded for the application. IMS can then use that PSB to access the metadata in the IMS catalog. There is no need to specifically define the catalog database in your PSB.

Direct catalog interface

An application user can issue a DL/I call by using the PCB that directly references the IMS catalog DBD if they need to access the metadata.

The application operates as a normal DL/I application. It can, for example, use the new GUR DL/I call to read the entire record (root and all the child segments) from the IMS catalog.

Indirect catalog interface

The indirect catalog interface is used when a process needs access to the metadata, but is not accessing the IMS catalog directly. The application processes as usual and can issue DL/I calls with a PSB that does not reference with the catalog DBD. When a process needs access to the metadata (in the past, the Java class from the DLIMODEL utility was used), IMS dynamically attaches a PSB with a catalog PCB. Figure 8-9 on page 303 shows this access.

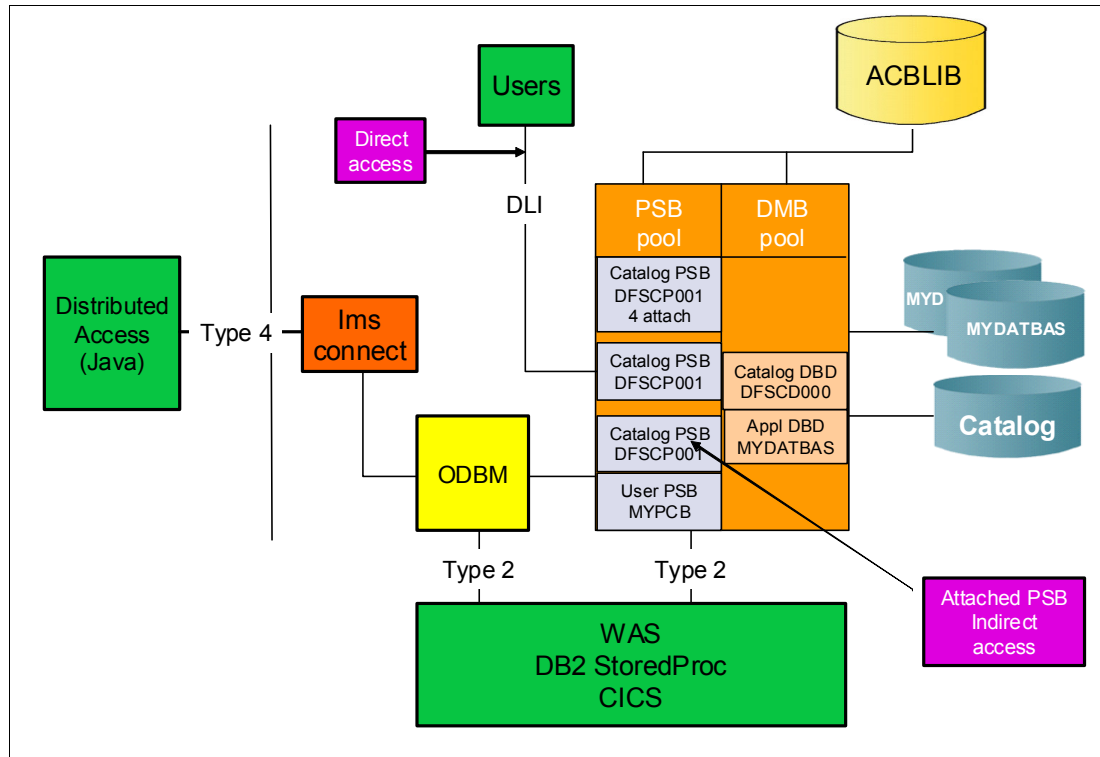


Figure 8-9 IMS catalog access

The catalog PCB is used by the DL/I engine without any external effects on the calling application. The normal mode of operation is that you are reading only the IMS catalog with a PROCOPT=G (read with integrity). The name of the main catalog PCB is DFSCAT00.

For message processing program (MPP) applications (running in an online IMS), the dynamic attach of a PSB that contains the IMS catalog PCB occurs at the first reference, that is, during the first DL/I call. The database is accessed with deferred scheduling, so the scheduling occurs only when the first call is made that needs access to the IMS catalog. The IMS catalog database availability does not affect application availability. If the IMS catalog is offline, you can get an abend (U3303), but that does not cause the application to terminate.

In a DL/I batch environment, the dynamic attach of the IMS catalog PCB occurs during the batch initialization. Scheduling is done only once for the batch when the region starts. A batch job loads the IMS catalog PSB for later reference if the catalog is enabled in the DFSDFxxx PROCLIB member.

8.4.4 SSA enhancements

The new SSA format, *get by offset*, allows a new search function by offset and length, instead of field name. Fields are no longer required to be defined in the DBD:

- ▶ Support is added for DFSDDL0 and IMS REXX.
- ▶ Performance is the same as a non-key field search.
- ▶ IMS scans the database to look for matches.
- ▶ SSA contains offset and length followed by operator and value.

In Figure 8-10, an SSA search on the field is issued, although the type is not known by the DBD.

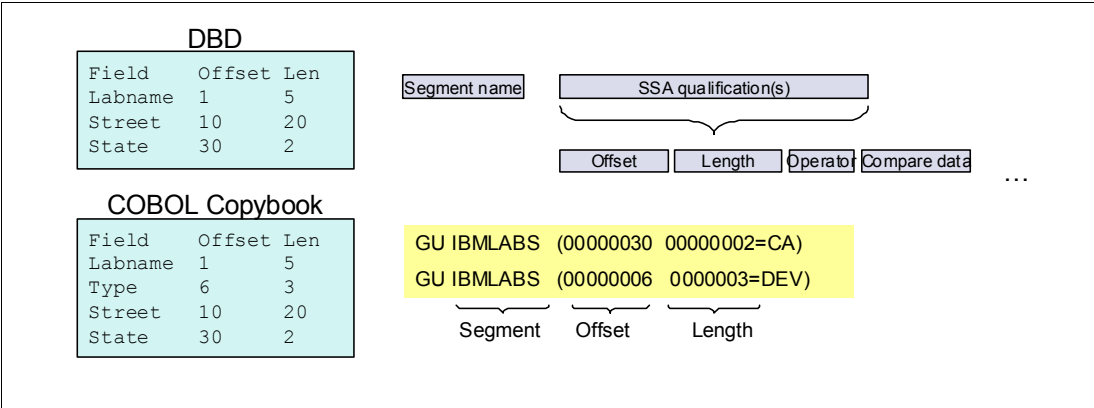


Figure 8-10 Using get by offset

Field level sensitivity allows the user to change the layout of the segment that is returned in the I/O area. The fields can be reordered, omitted, or spaces can be added between, as shown in Figure 8-11. The new SSA qualifications can be used in combination with the existing SSA qualification format. For example, the following SSA combination is valid for the fields: labname and type.

GU IBMLABS*0 (LABNAME =SVL &00000006 00000003=DEV)

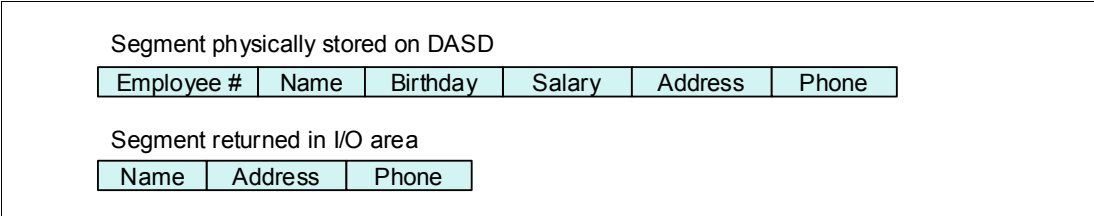


Figure 8-11 Field sensitivity

In this request, the SSA offset is relative to the physical starting position in the segment that is visible to your program. If you attempt to search for fields that are outside of the sensitive area (that is, past the end of the segment), IMS returns an AK status code. If a segment is not found that matches the SSA qualifications, a GE or GB status code is returned.

8.5 The role of the IMS Enterprise Suite Explorer for Development

The IMS Enterprise Suite Explorer for Development (*IMS Explorer*) is an Eclipse-based graphical tool that simplifies many IMS application development tasks. The tool simplifies tasks such as updating IMS database and program definitions, and by using standard Structured Query Language (SQL) to manipulate the IMS data.

You can use the IMS Explorer graphical editors to import, visualize, and edit IMS database and program definitions. You can also use the IMS Explorer to easily access and manipulate data that is stored in IMS by using standard SQL.

The IMS Explorer can also directly import DBDs and PSBs, or can obtain existing catalog information from the IMS catalog through a type-4 connection.

The population of the IMS catalog is done from the PSB and DBD sources. Additional **DBDGEN** statements and additional attributes on the **FIELD** macro can provide complete metadata information.

Additional metadata can be collected from COBOL copybooks and PL/I or C include members. After updating the metadata information in the PSBs and DBDs in the IMS Explorer, those sources can be sent to the host for the GEN process. This process causes the metadata to be populated into the IMS catalog.

Figure 8-12 shows how the IMS Explorer is able to interact with the IMS catalog directly through a type-4 driver connection. Figure 8-12 also shows an interaction with the catalog indirectly through a File Transfer Protocol (FTP) import and export.

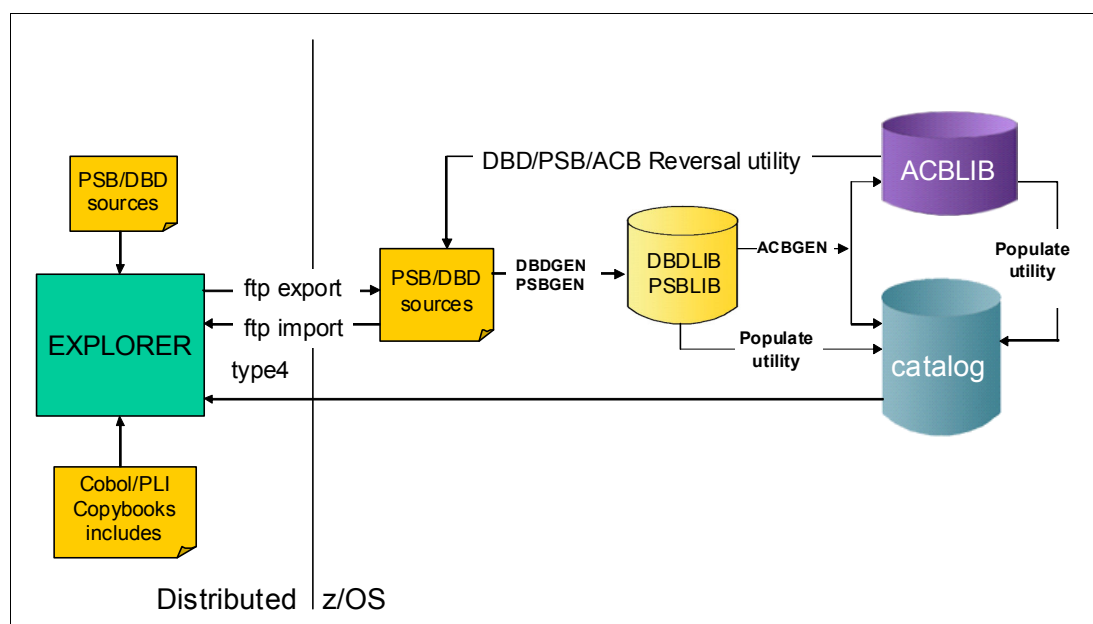


Figure 8-12 IMS Explorer and catalog

A direct update from the IMS Explorer is not available. An intermediate application control block generation (ACBGEN) with population of the catalog is required.

The DLIModel utility is a helpful tool for visualizing the structure of an IMS database. The tool documents the details of the DBD and PSB sources. The DLIModel utility is required to produce the `com.ibm.ims.db.DLIDatabaseView` class for accessing the IMS databases from Java applications on the remote and local locations. This access is through the type-4 and type-2 drivers. Both the traditional DL/I SSA and JDBC access can be used.

The DLIModel utility was not changed in Enterprise Suite V2, but its functions were integrated into the IMS Enterprise Suite Explorer for Development. The IMS Explorer uses the centralized IMS catalog on IBM z/OS. The metadata that is available in the DatabaseView class is integrated in the IMS catalog. The concept of the IMS catalog makes the metadata more dynamic and shared. Its implementation avoids the need for distributing the class to all sites where it might be used in Java programs with DL/I (or JDBC) access. The access is through the IMS database type-2 and type-4 Universal drivers. The drivers were adapted to provide this function.

The following features summarize the IBM Enterprise Explorer:

- ▶ Incorporates DLIModel functionality.
Support for migration of DLIModel projects is provided.
- ▶ Provides the following graphical editors for development and visualization:
 - Data Base Description (DBD)
 - Program Specification Block (PSB)
- ▶ Shows a relational view of IMS data with graphical assistance to build SQL statements.

The IMS Universal JDBC type-4 connectivity extracts DBD and PSB information for local enhancement.

Input for the IMS Explorer

Section 8.5, “The role of the IMS Enterprise Suite Explorer for Development” on page 304 explains how the input for the IMS Explorer is taken from local PSB and DBD sources.

The IMS Explorer can also take input from other sources:

- ▶ Import of the DLIModel utility project
- ▶ Remote import
- ▶ From the IMS catalog

All information about the PSBs and DBDs is consolidated into the IMS catalog. The IMS Explorer, using the new type-4 driver, can extract PSB and DBD information from the catalog to be updated locally. After the update, this information is sent back to the consolidating host by FTP. Figure 8-13 shows the Export window.

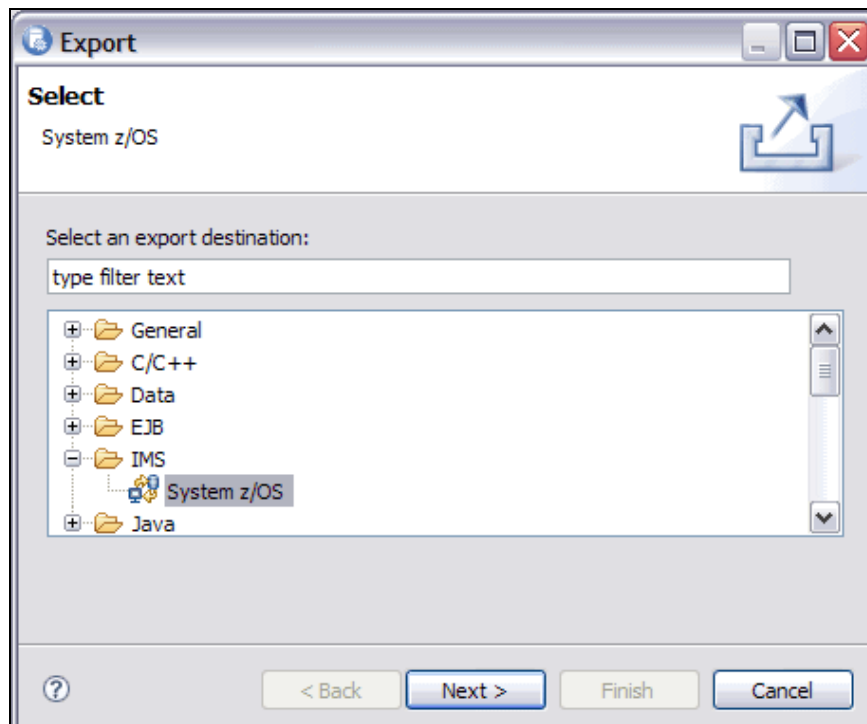


Figure 8-13 Export to the z/OS host with FTP

An ACBGEN of the changed PSB or DBD is needed to update the ACBLIB and populate this updated information to the catalog.

IMS Enterprise Suite Explorer and the IMS catalog

The IMS catalog is a PHIDAM database that contains trusted metadata for IMS databases and applications. All the information that is contained in the ACBLIB run time (from the DBDLIB and PSBLIB) is available to users in the IMS catalog.

The IMS Explorer is an important tool for using and updating the information in the catalog. The IMS Explorer can access information from the catalog database, work with it, update it, and send it back to the IMS on z/OS for consolidation in the catalog.

The following data connections can be used between the IMS Explorer and catalog information:

- ▶ From z/OS:
 - FTP import from the DBD and PSB sources
 - Through type-4 driver direct access to the catalog
- ▶ To z/OS:
 - FTP export to the DBD and PSB sources

8.5.1 Extending IMS database definitions with the IMS Explorer

This section describes the extension of an IMS DBD with the IMS Explorer for Development. Using the IMS Explorer, you can quickly and easily add metadata to the IMS databases. You can do this from the COBOL copybooks or PL/I include members that are used by application programs.

8.6 Using IMS Explorer to capture IMS metadata

The IMS Explorer facilitates the capture of more metadata for an IMS database than was previously stored in the IMS DBD. With IMS Explorer, the layout of the data in the database segments can be captured and added to the DBD. The enhanced DBD can then be generated back on the host system and included in the catalog. This section describes how to use the IMS Explorer for Development to capture the extra metadata information from the COBOL copybooks. This section also describes how the enhanced DBD is ready for generation on the host system.

Before you start, you must install the IMS Explorer for Development. The code is available as part of the IMS Enterprise Suite Version 2, which is freely downloadable from the IMS home page (<http://www.ibm.com/ims>). There are several runtime versions of this code. The code that is needed for this task is the *shell-sharing* version. This version installs with the IBM Installation Manager as an extension to IBM Rational Developer for zSeries or IBM Rational Developer for zEnterprise®.

Complete the following steps:

1. Run Rational Developer for zEnterprise (or Rational Developer for zSeries), and open the IMS Explorer perspective, as shown in Figure 8-14.

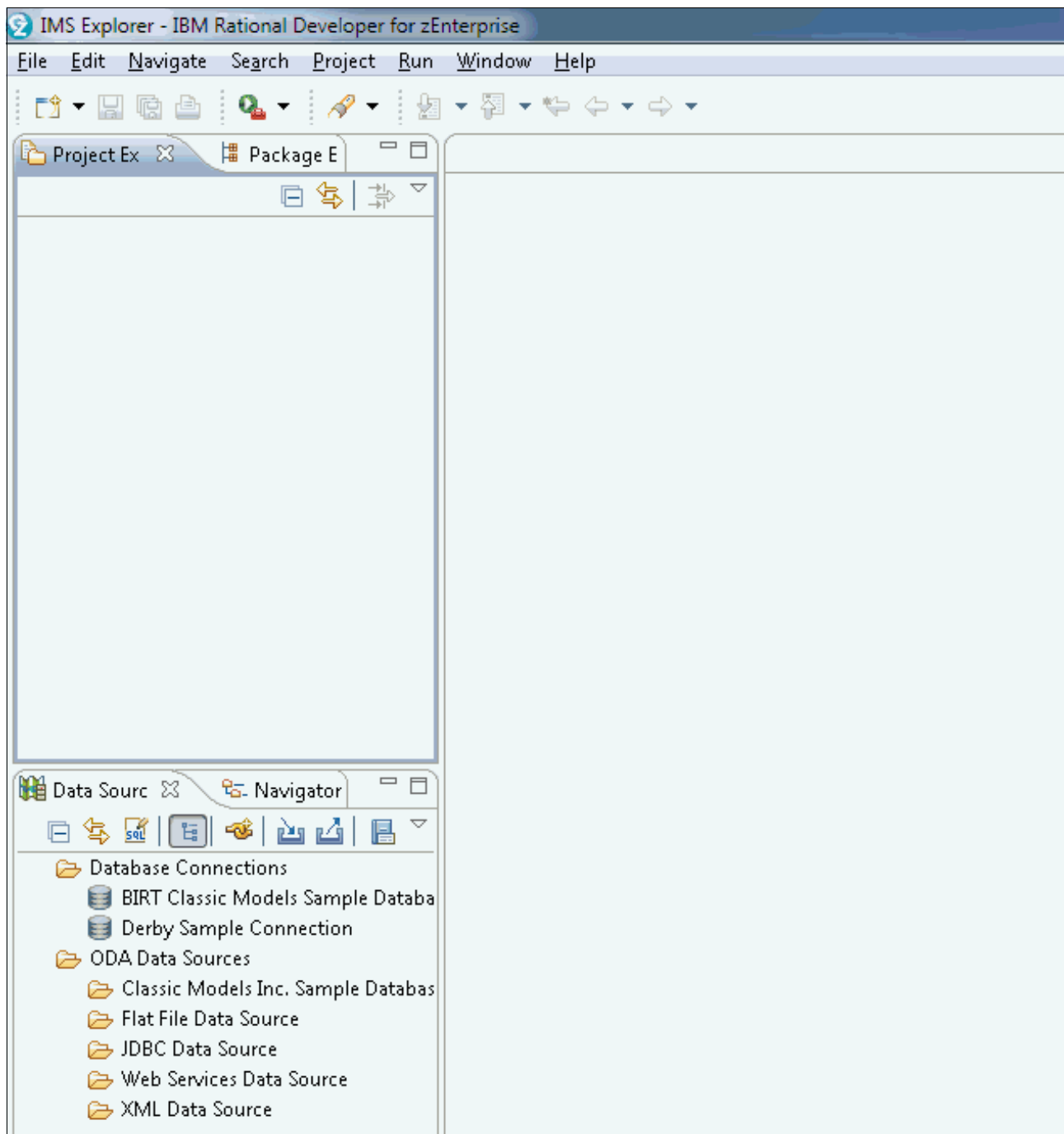


Figure 8-14 IMS Explorer perspective in Rational Developer for zEnterprise

2. To create an IMS Explorer Project, right-click in the Project Explorer window (upper left), and select **New** → **Project** to start the Create a New Project wizard (Figure 8-15 on page 309).

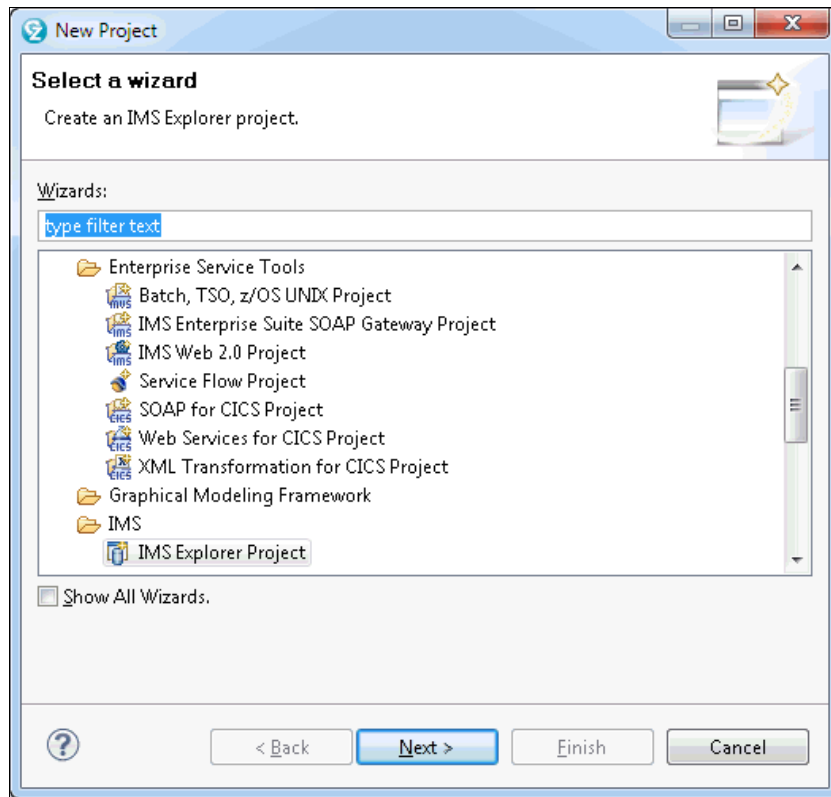


Figure 8-15 Create a New Project wizard

3. The wizard is the IMS Explorer Project wizard. Select this one, and click **Next**, as shown in Figure 8-16.

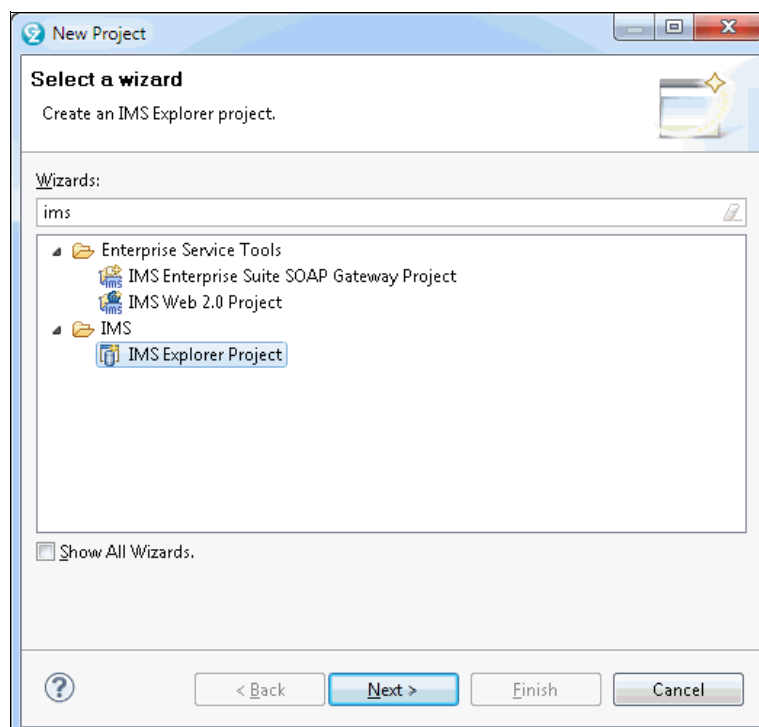


Figure 8-16 IMS Explorer Project

4. Enter a name for the IMS Explorer Project, as shown in Figure 8-17.

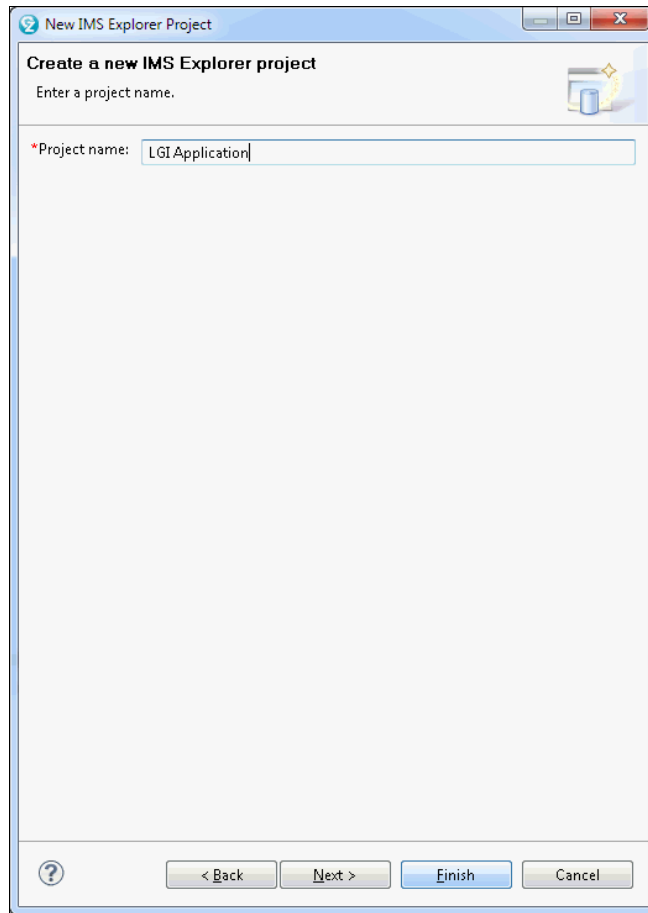


Figure 8-17 Name the IMS Explorer Project

When the wizard completes, Rational Developer for IBM System z shows the named project in the Project Explorer window (see Figure 8-18).

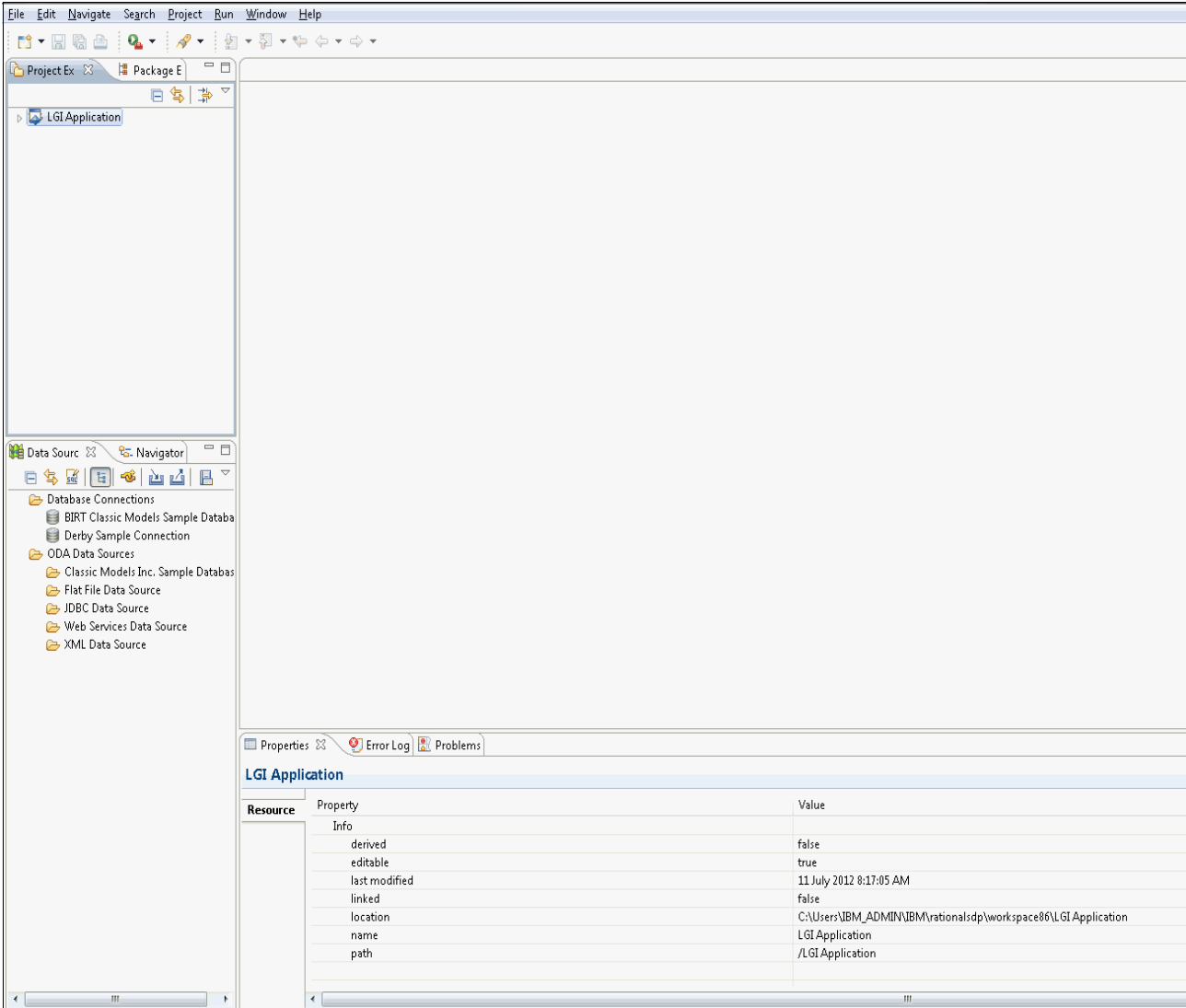


Figure 8-18 The LGI Application IMS Explorer Project

Next, you must import the DBD for the databases for which you are going to capture the metadata. To import the DBD, complete the following steps:

1. Right-click the Explorer Project (in this case, the LGI Application), and select **Import** (Figure 8-19).

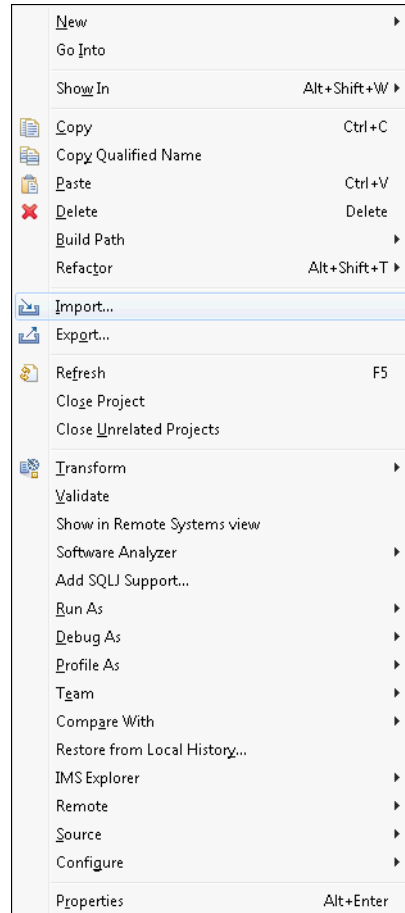


Figure 8-19 Options for an IMS Explorer Project

2. Select the **IMS Resources** option, and click **Next**, as shown in Figure 8-20.

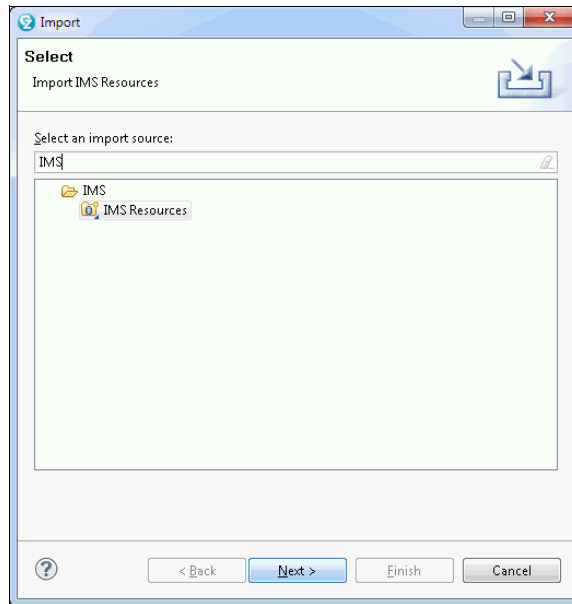


Figure 8-20 Select an import source

3. Enter the name of the project, as shown in Figure 8-21.

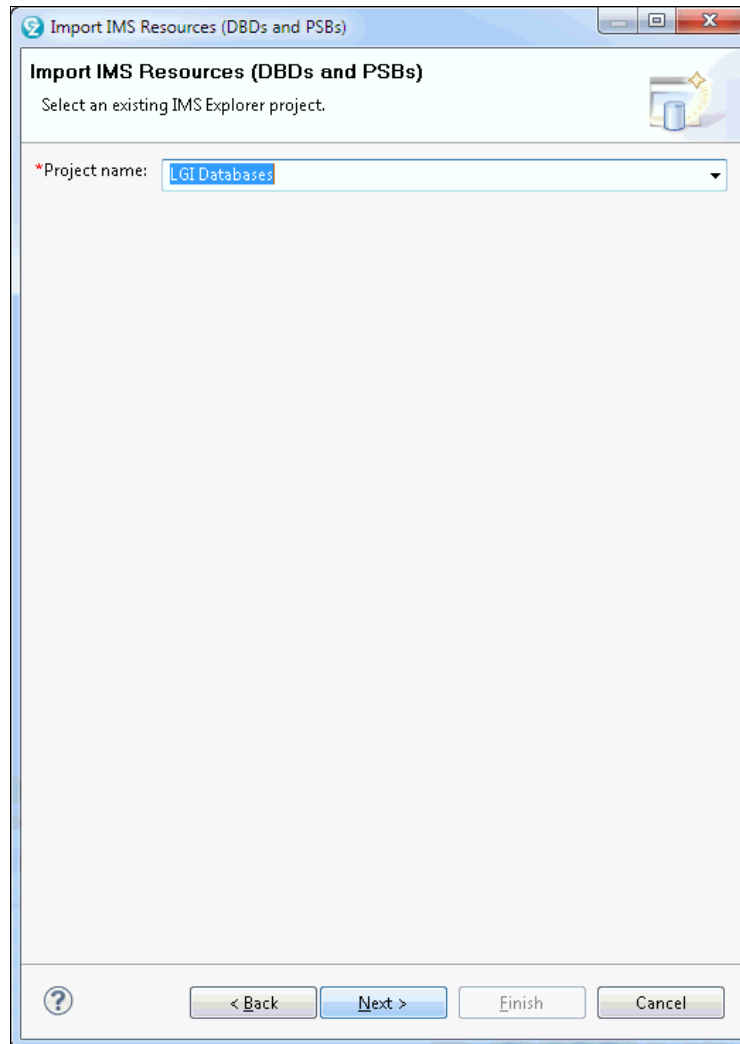


Figure 8-21 Provide a name for the IMS Resources to be imported

4. In Figure 8-22, select a location from which the resources are imported. The location can be a file on your local workstation, a file on a host in which you are connected, or an IMS catalog. Click **Next**.

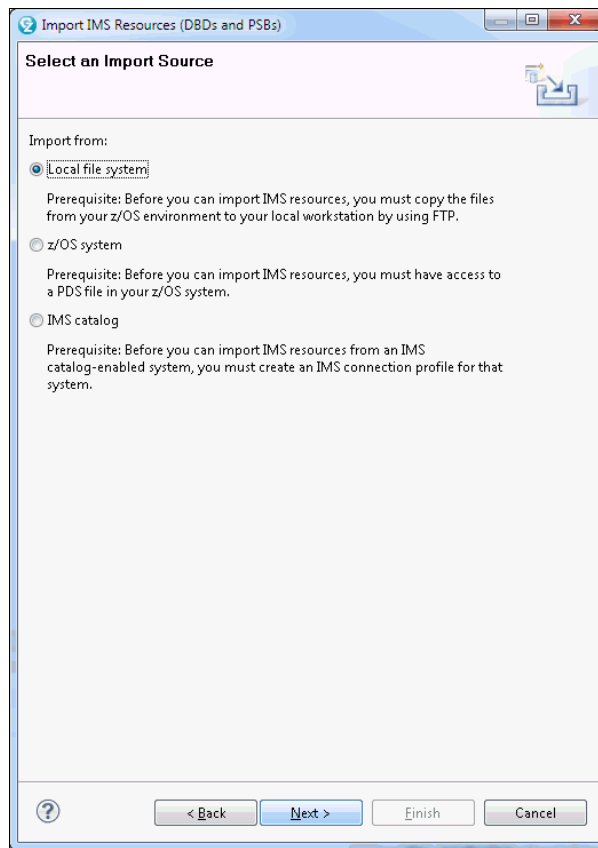


Figure 8-22 Select the Import Source of the source to be imported

5. You can now select the files from the source that you want imported into your project. Although there are separate radio buttons on this window for DBDs and PSBs, IMS Explorer can identify the resources in the source file selected. It then lists them in the appropriate list in this window. IMS Explorer also understands a file of JCL containing multiple DBDs, PSBs, or a mixture of both, which simplifies the import process.

The first resources to be imported here are DBDs. Click **Add DBD**, as shown in Figure 8-23.

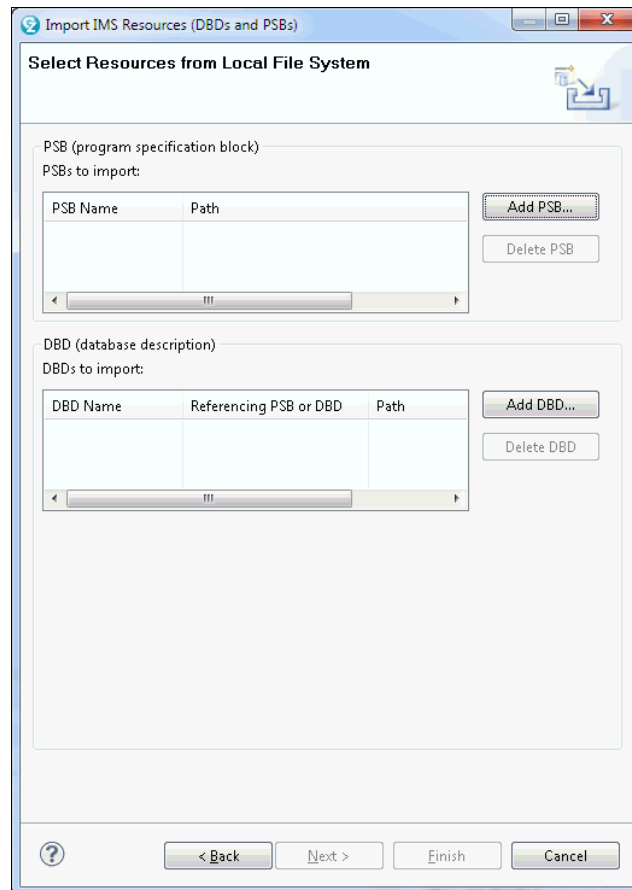


Figure 8-23 Select resources from the local file system

6. Select the file that contains the DBDs to be imported, as shown in Figure 8-24. In this example, multiple DBDs are in a JCL file that was previously downloaded to the local workstation. IMS automatically filters out the JCL and imports each of the DBDs in the file.

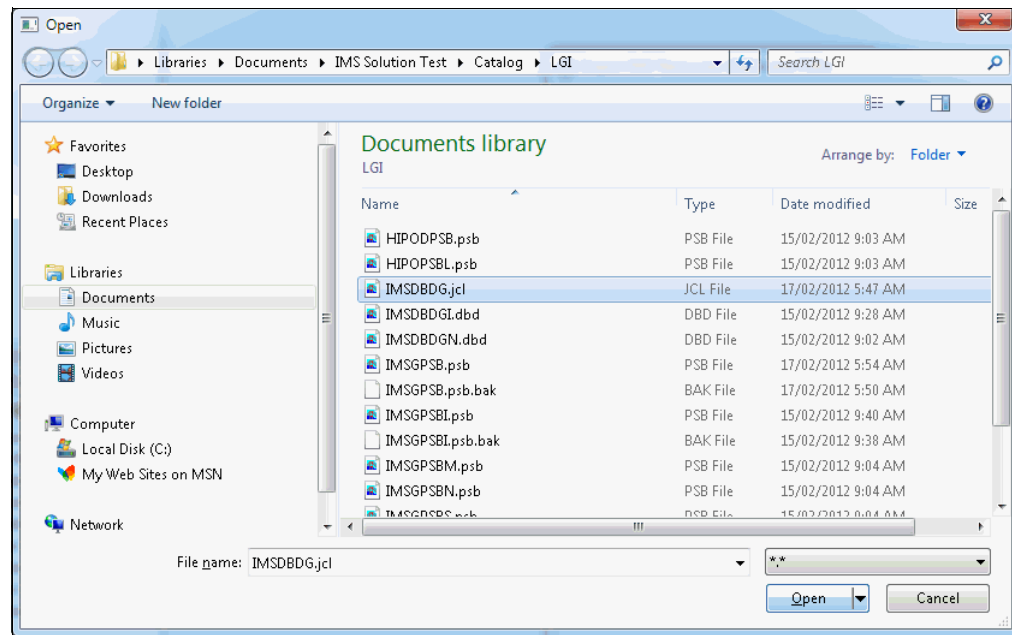


Figure 8-24 Select the file that contains the resources to be imported

The Import IMS Resources window shows the two DBDs (S2U1DBD and S2U1DXD) that were in the single JCL file that was imported. Any of these DBDs can now be deleted from the import list, if you want.

7. To add PSBs to this import, click **Add PSB**, as shown in Figure 8-25.

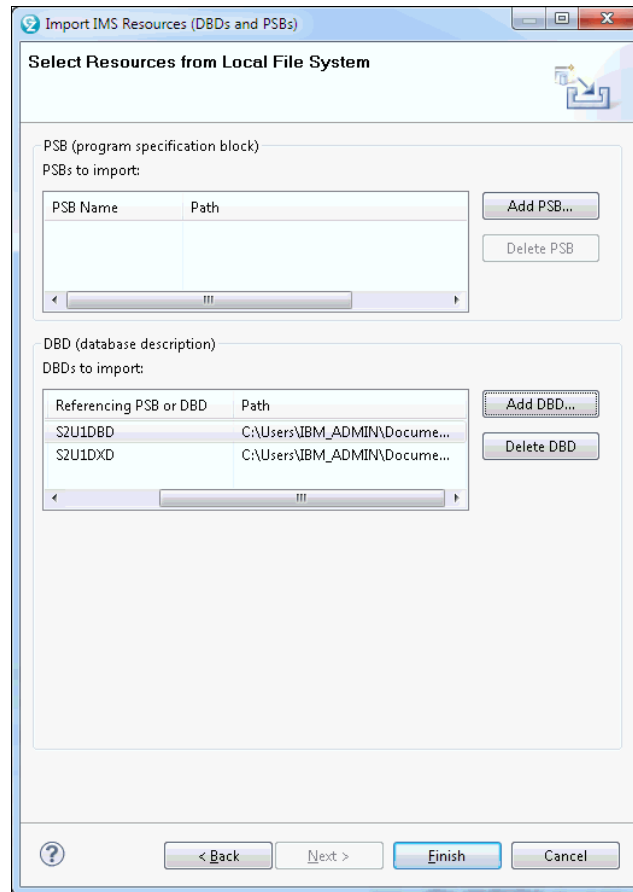


Figure 8-25 Keep selecting more files until you are done

As with the DBDs, a single file can contain multiple PSBs, and possibly the JCL that is used to generate them. IMS looks for each of the PSBs and DBDs in the file that is selected and adds each of them to the import selection.

8. Select the file that contains the PSBs that you want to import, and click **Open**, as shown in Figure 8-26.

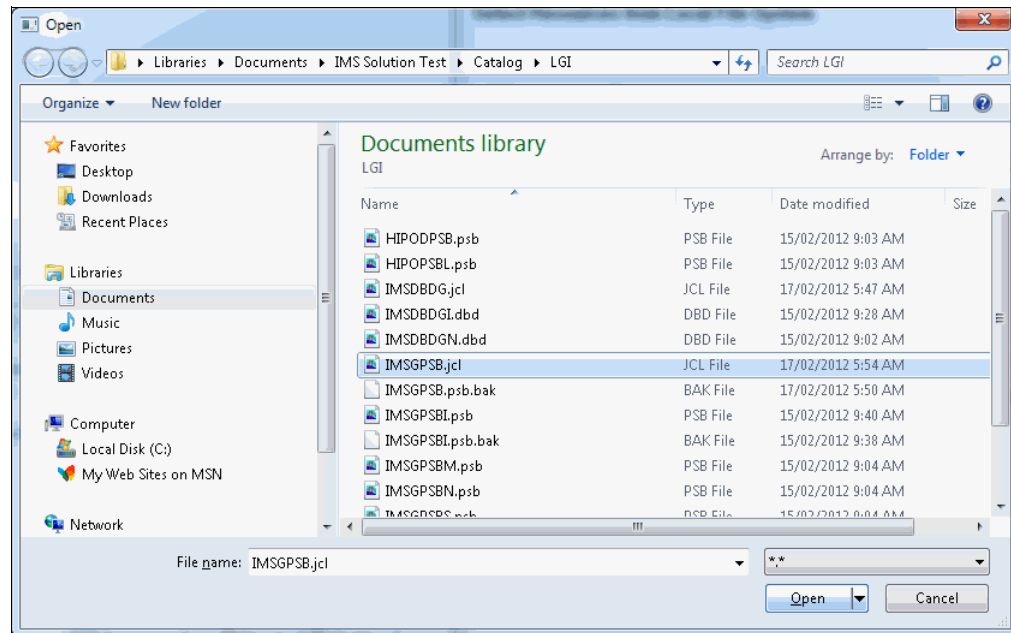


Figure 8-26 Selecting the PSB source to be imported

The Import IMS Resources window shows the two PSBs that were imported from the single file that was selected. Any of these DBDs and PSBs can now be removed from the import list by clicking **Delete DBD** and then by clicking **Delete PSB**.

The IMS Explorer also shows any DBDs that still must be imported to satisfy the databases that are referenced in the PSBs to be imported. In this example, the two PSBs reference only the databases to be imported.

9. You are now ready to import these DBDs and PSBs into your project. Click **Finish**, as shown in Figure 8-27.

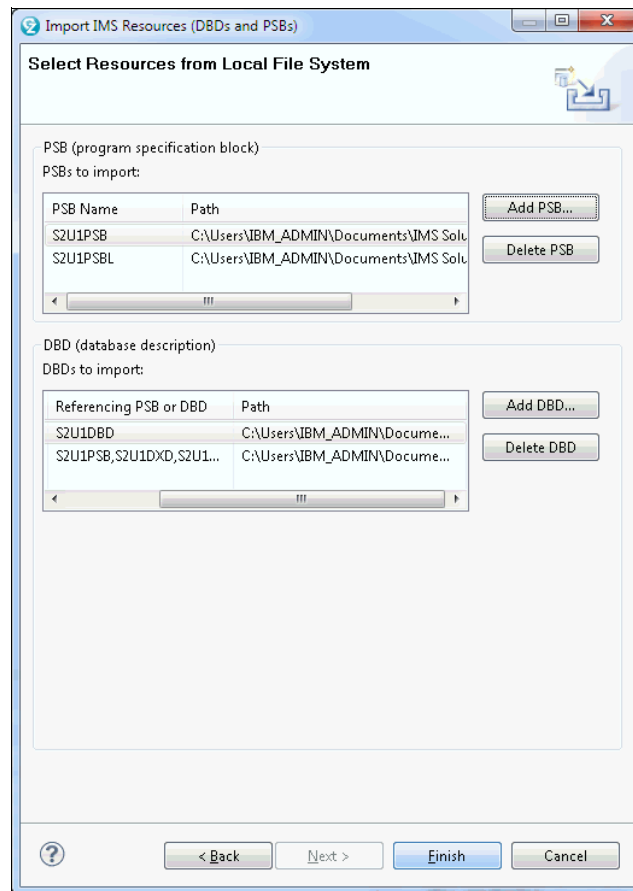


Figure 8-27 Lists of the DBD and PSB resources to be imported

The LGI Databases project now includes the DBDs and PSBs that you imported and several other artifacts that are needed later. At any time, you can refer to this project to see the DBD or PSB source that was imported. The project also includes the generated source for the DBDs and PSBs. At this stage, however, the generated source is the same as the imported source. The sources are the same because you did not yet add any of the metadata from the COBOL copybooks or PL/I include members.

To see the current layout of a database, click the arrow at the left of the DBD to expand the list of DBDs in your project, as shown in Figure 8-28 on page 321.

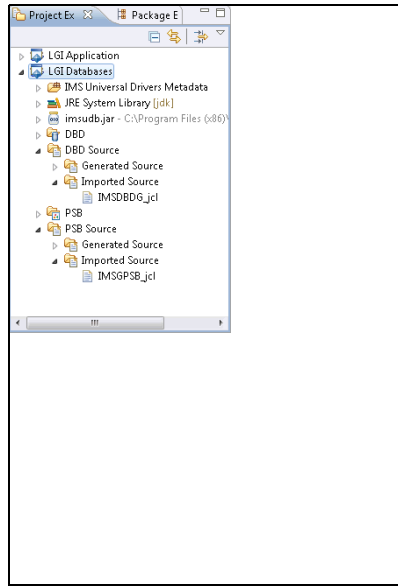


Figure 8-28 *LGI Databases project showing the DBD and PSB resources that are imported*

You can now see the two DBDs that are imported into your project, S2U1DBD.dbd and S2U1DXD.dbd. Double-click the S2U1DBD.dbd option, as shown in Figure 8-29, to see the view of this database.

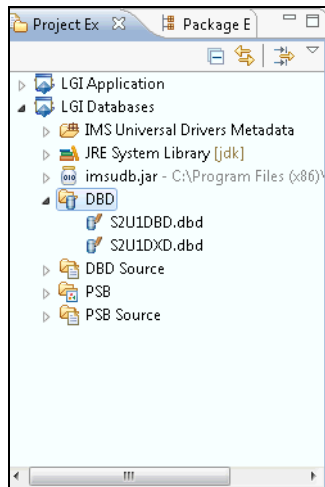


Figure 8-29 *Expanding the DBD Resources view*

You can now see the graphical layout of the database, as shown in Figure 8-30. The segments in the S2U1DBD database are shown. The relationships between these segments, and the fields that are defined to each of the segments, are also shown.

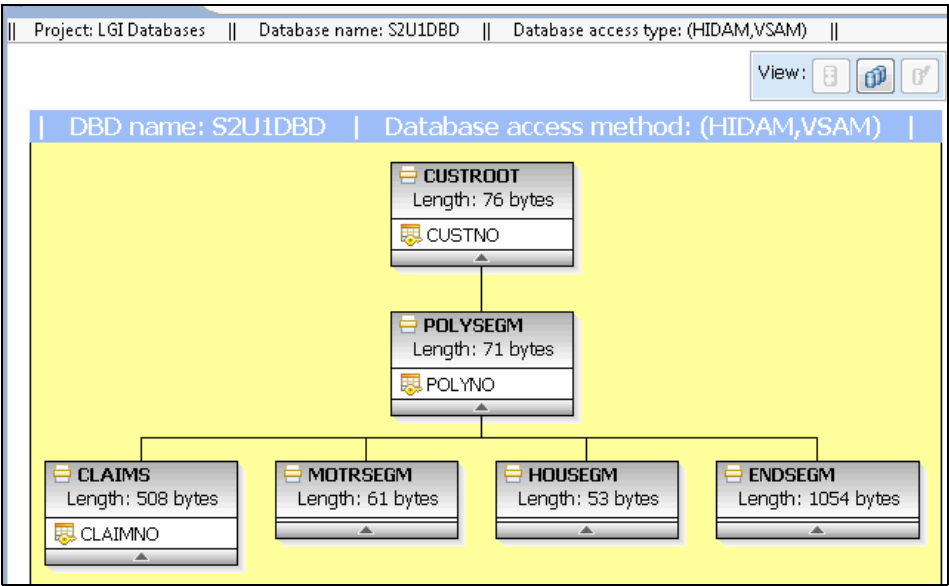


Figure 8-30 The IMS Explorer view of an IMS database

From this window, you can import the rest of the metadata for this database into this IMS Explorer project. To do this import, complete the following steps:

1. Right-click the CUSTROOT segment in the graphical view, and select **Import COBOL or PL/I Data Structures**. This action opens the window that is shown in Figure 8-31.

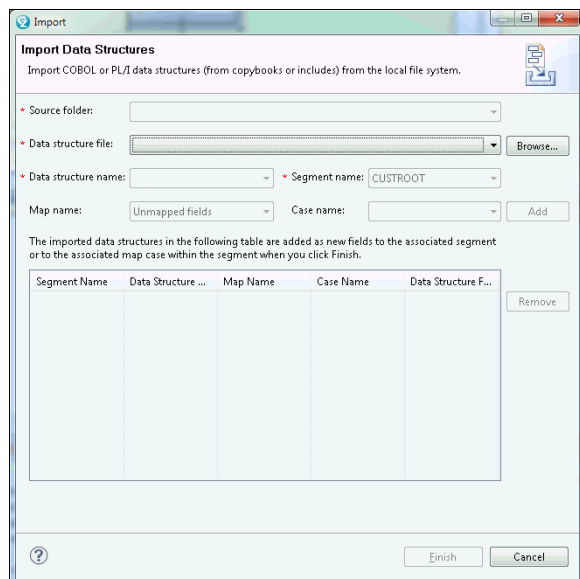


Figure 8-31 Importing database metadata from COBOL or PL/I

To select the file that contains the COBOL or PL/I definition for the segment, click **Browse**, as shown in Figure 8-31.

2. Select the file that contains the COBOL or PL/I definition for the custroot (client) segment, and click **Open**, as shown in Figure 8-32.

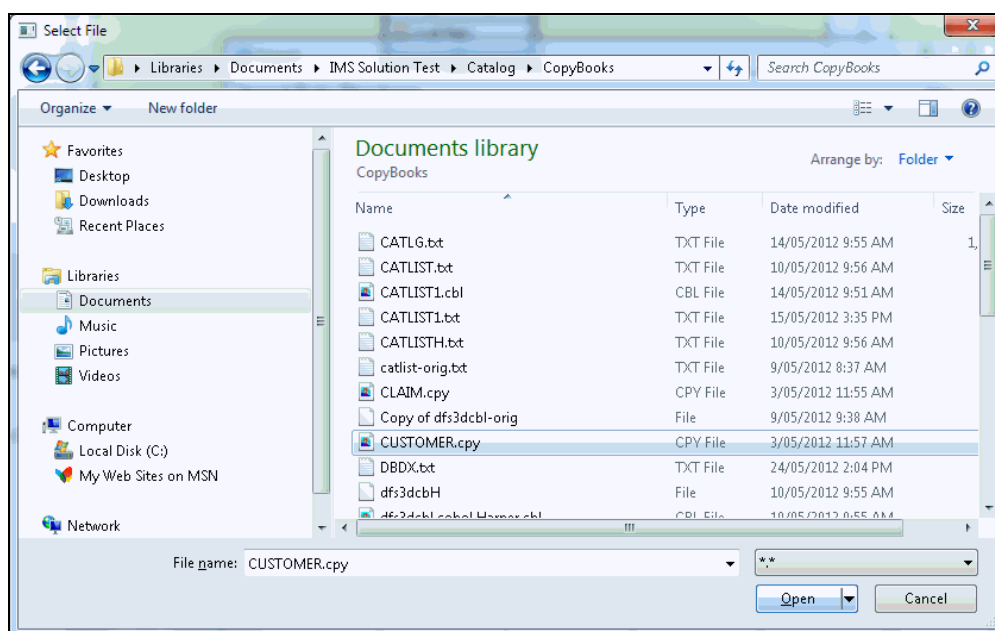


Figure 8-32 Select the file that contains the copybook or include member

Figure 8-33 shows the window that displays when you select **Open** (see Figure 8-32 on page 323). Figure 8-33 shows the Import Data Structures window. This window is where you identify the data structure to be imported for a database segment.

Import Data Structures
Import COBOL or PL/I data structures (from copybooks or includes) from the local file system.

* Source folder: C:\Users\IBM_ADMIN\Documents\IMS Solution Test\Catalog\CopyBooks

* Data structure file: CUSTOMER.cpy

* Data structure name: CustomerSegment * Segment name: CUSTROOT

Map name: Unmapped fields Case name:

The imported data structures in the following table are added as new fields to the associated segment or to the associated map case within the segment when you click Finish.

Segment Name	Data Structure ...	Map Name	Case Name	Data Structure F...
CUSTROOT	CustomerSegm...	Unmapped fields		C:\Users\IBM_A...

Figure 8-33 Identify the data structure to be imported for a database segment

3. In the COBOL copybook that you selected, there is only one definition layout. Click **Add**, as shown in Figure 8-34, to add this structure as the metadata for the CUSTROOT segment.

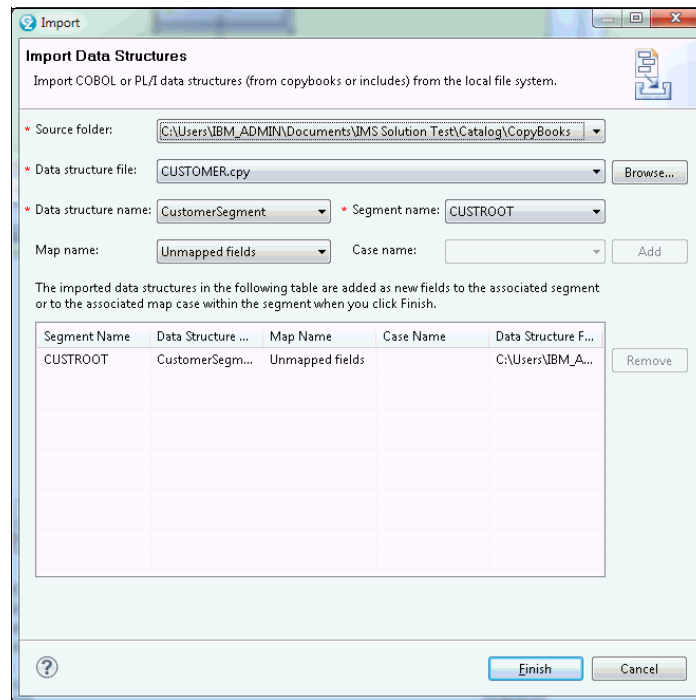


Figure 8-34 Importing the metadata for a database segment

4. Click **Finish** to import this segment layout into the database definition for the CUSTROOT segment. Figure 8-35 shows the expanded definition for this segment in the database.

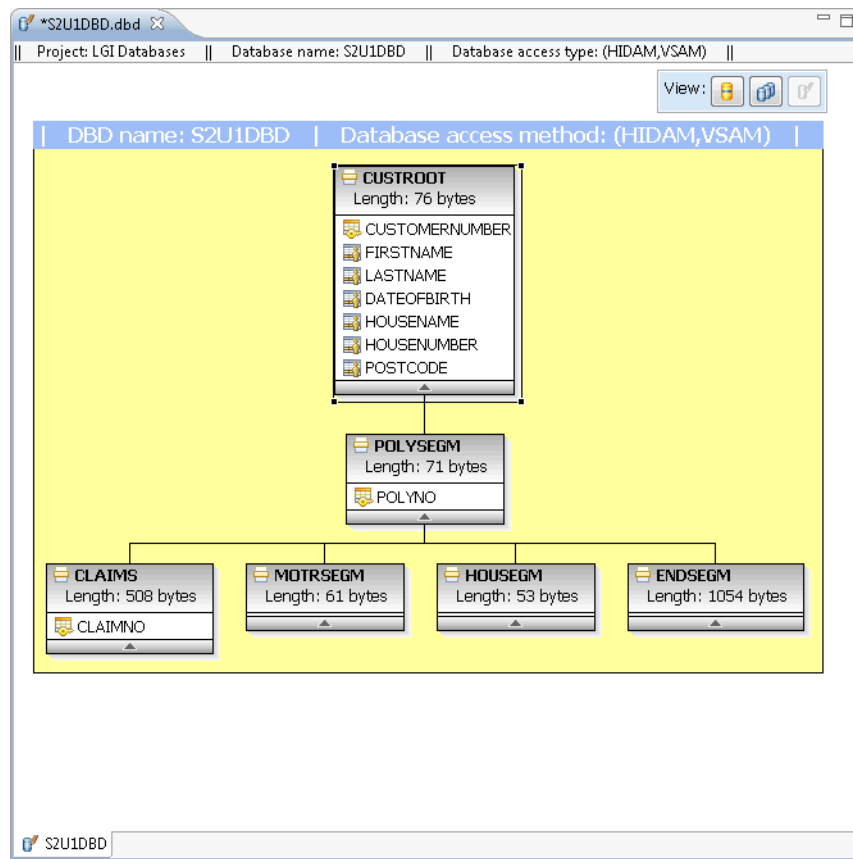


Figure 8-35 IMS Explorer database view with imported metadata for one segment

The same process (right-click the segment, import the COBOL or PL/I copybook, click **Add**, and then **Finish**) is used to import the metadata for each of the other segments in the database.

Figure 8-36 shows the database layout with definitions for each of the segments in the database. The asterisk at the left of the database name (*S2U1DBD.dbd) indicates that the database definition is not yet saved. Click **File** → **Save** to save the enhanced database definition to the *LGI Database Project*.

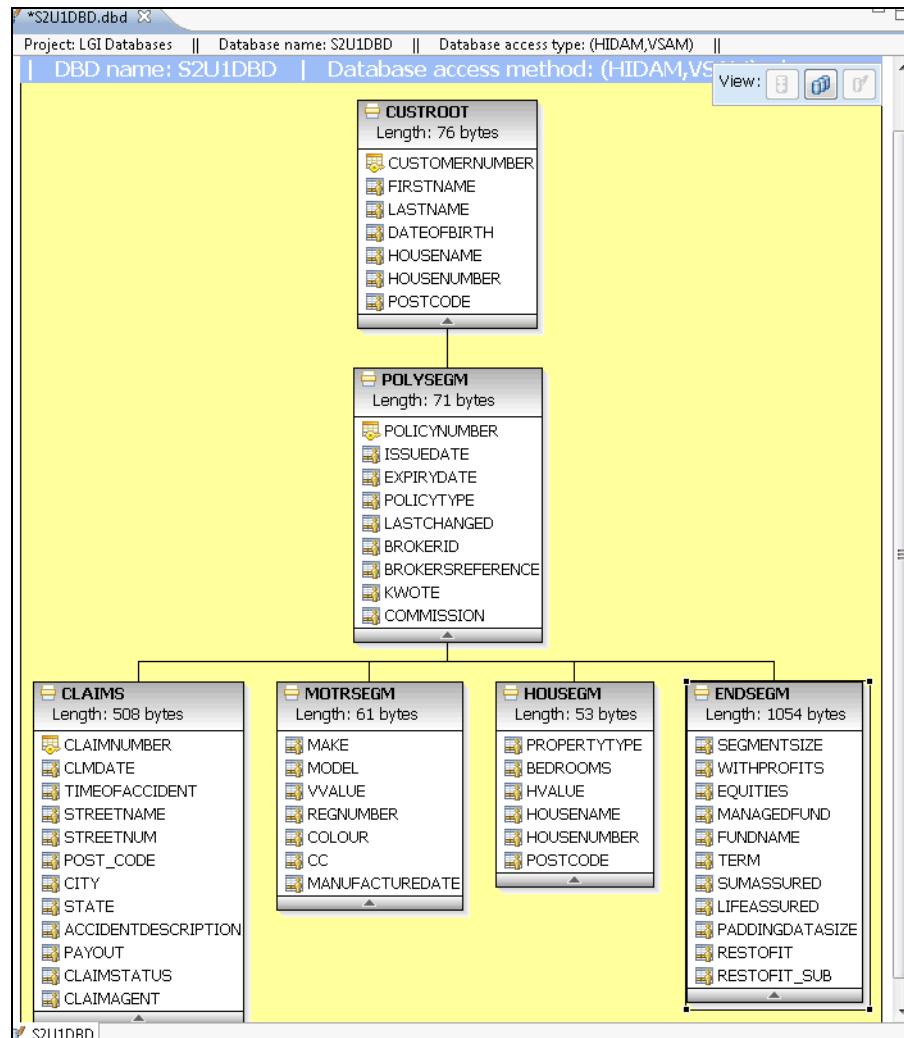


Figure 8-36 Explorer database view with imported metadata for all segments

You captured the metadata for the layout of each of the segments in your database. If you click the expand arrow for DBD Source and Generated Source in your project, the IMS catalog-enabled DBD source files heading opens. Figure 8-37 shows the enhanced definitions for the databases in your project.

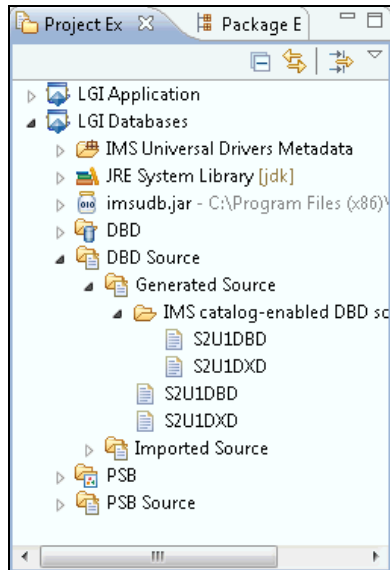


Figure 8-37 The catalog-enabled DBD source

You captured the expanded metadata for these databases. A snippet of the expanded DBD definition for this database is shown in Figure 8-38.

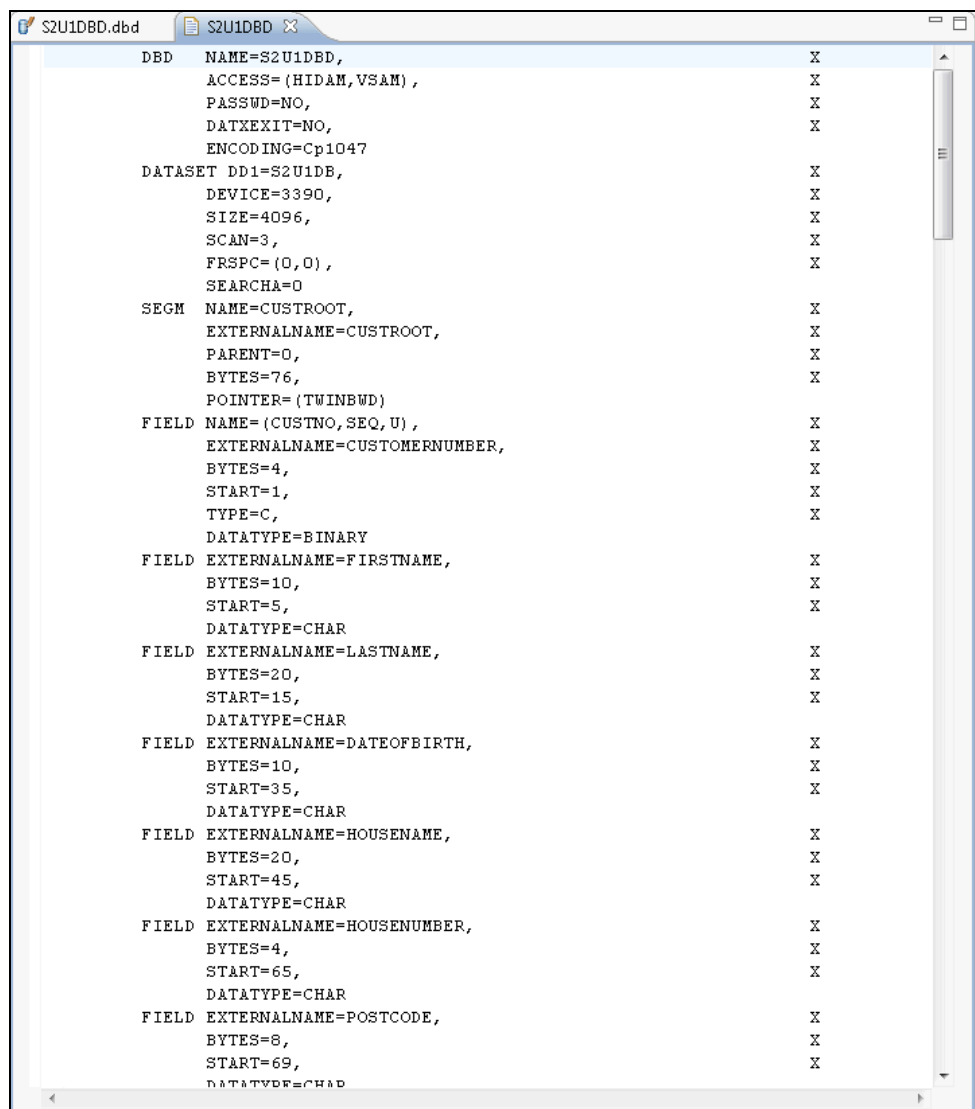


Figure 8-38 Sample of a DBD with catalog-enabled source

From here, you can use existing IMS database definition processes to include the enhanced database definition into your IMS system. The following steps describe how to include the definition in your system:

1. Copy the expanded DBD source to the host.
There are several ways to do this copy. Recent versions of IBM Personal Communications include an FTP client with a graphical user interface (GUI). A simple file transfer is also available through your 3270 emulator.
2. Generate these DBDs with standard DBDGEN processes.
3. Perform an ACBGEN and Catalog Populate utility execution.
The updated database description must be added into your ACB library and also to the catalog. This process can be done in a single utility execution or separately.

The database metadata is now captured in the catalog. Existing applications continue to access their databases as they always have. New Java applications can use the catalog definitions to simplify the JDBC access to the same IMS databases.

8.7 Enhancements to the IMS Universal drivers

All IMS Universal drivers are enhanced to use the IMS catalog and provide the following features:

- ▶ There is direct access to IMS metadata in the catalog
- ▶ The drivers no longer require the separate Java metadata class.
- ▶ The drivers are no longer file-system dependent for metadata: There is virtual deployment support.
- ▶ The metadata is trusted and up-to-date.
- ▶ The Application Development community can access any IMS database that is defined in the catalog.
- ▶ There is new complex and flexible data-type support.

The following two sections describe the changes to the IMS drivers:

- ▶ Access to the IMS databases from Java
- ▶ Using the metadata information in the DL/I access

8.7.1 Access to the IMS databases from Java

You must establish a connection before any data can be accessed from DL/I databases. Traditionally, access with IMS and DLI requires a PSB; this statement is also true for Java. Traditional languages, such as COBOL and PL/I, work with a segment content, which is accessed by the SSA protocol. Java supports the following two ways to access DL/I databases through the Universal drivers (type-4 for distributed and type-2 for local):

- ▶ SSA oriented (for DL/I clients)
- ▶ JDBC clients

JDBC requires the additional support of metadata for the accessed information. In previous releases, this support was provided by DatabaseView classes that are generated by the DLIModel utility. IMS V12 added a more centralized approach by putting the metadata in to a catalog. The usage of metadata information offers many interesting access perspectives to the Java language (case mapping, structure selects, and so on) that are not available in other languages.

The metadata support can come from the DLIModel or the IMS catalog. This choice needs to be made at connection time, as shown in Figure 8-39.

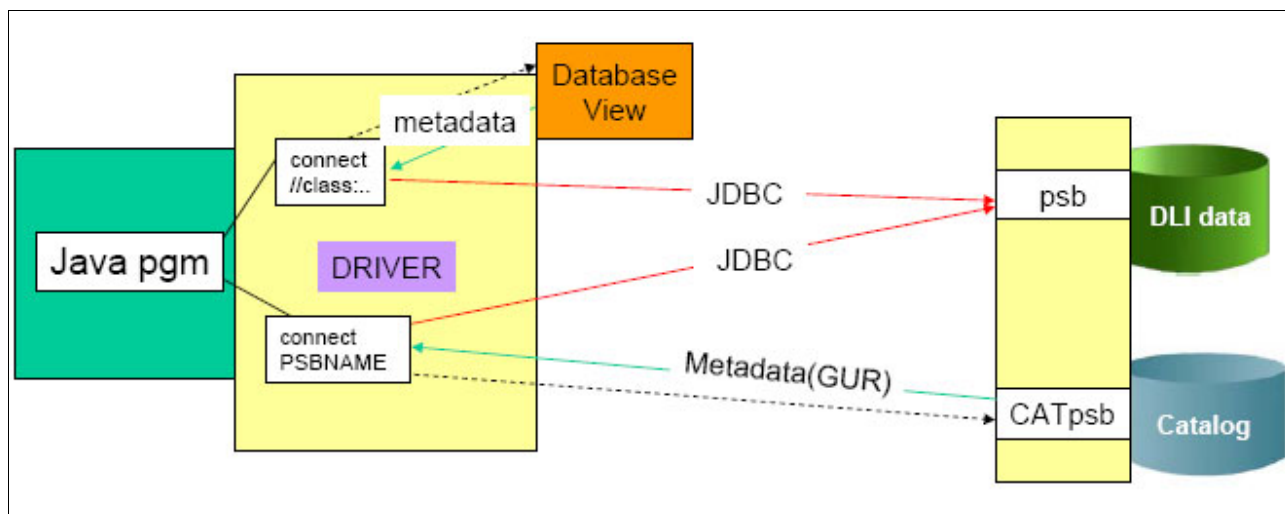


Figure 8-39 Connection of a Java program through the drivers

Two elements are required before you are able to access the DL/I databases:

1. A PSB (for language Java) must exist.
2. A DatabaseView class or an IMS catalog for obtaining metadata:
 - A DatabaseView class must be generated by the DLIModel utility.
 - Metadata information is available in the catalog.

Connecting to IMS for use as a JDBC client

Example 8-19 shows a code excerpt of how to obtain a type-4 connection (with an IP address and port number) with IMS for a JDBC client.

Example 8-19 Obtain a connection with IMSDataSource for use with JDBC

JDBC connection:

```
import java.sql.Connection;
import java.sql.SQLException;
import com.ibm.ims.jdbc.IMSDataSource;
...
IMSDataSource ds = new IMSDataSource();
Connection conn = null;

ds.setDatastoreName(alias); // IMS alias name defined in ODBM
ds.setDatabaseName(psbName);
// ds.setMetadataURL(url); ---> deprecated

ds.setUser(user);
ds.setPassword(password);
ds.setDriverType(driverType);

// optional settings
ds.setLoginTimeout(seconds);
ds.setDescription(description);
try {
```

```

        ds.setLogWriter(out); // set to a java.io.PrintWriter object
    } catch (SQLException e) {
        // handle exception
    }

    // JDBC type 4 driver specific settings
    if (driverType == IMSDataSource.DRIVER_TYPE_4) {
        ds.setSSLConnection(enableSSL);
        ds.setDatastoreServer(host); // IP address or DNS name of the
            // LPAR where ICON resides
        ds.setPortNumber(port); // ICON DRDA port number
    }

    try {
        // Establish a connection using the data source
        conn = ds.getConnection();

        // do some work here

        conn.close();
    } catch (SQLException e) {
        // handle exception
    }
}

```

Example 8-19 on page 331 shows the usage of `IMSDataSource`, which gets its properties through several setters and the `getConnection()` method to obtain the connection object.

As shown in Example 8-20 on page 333, the `setMetadataURL()` method on the `IMSDataSource` is now deprecated and replaced by `setDatabaseName(url)`.

The `setDatabaseName(url)` method sets the name of the target IMS databases to be accessed:

- ▶ If the metadata repository is the IMS catalog, the database name (url) is the name of the PSB that contains the databases to be accessed.
- ▶ If the metadata repository is the `DatabaseView` class, the database name is the fully qualified name of the Java metadata class that is generated by the `DLIModel` utility. In this case, the name must begin with `class://`.

JDBC client connection: For a JDBC client, the `Connection` object is the target for all other interaction with the DL/I databases.

Connecting to IMS for use as a DL/I client

Example 8-20 on page 333 shows an excerpt of the code that is required to obtain access to the DL/I data from a DL/I client through the PCB object. Through a DL/I client, access takes place as in the other languages, with PCB, SSAs, and function codes.

DLI connection:

```
import com.ibm.ims.dli.DLIException;
import com.ibm.ims.dli.IMSConnectionSpec;
import com.ibm.ims.dli.IMSConnectionSpecFactory;
import com.ibm.ims.dli.PCB;
import com.ibm.ims.dli.PSB;
import com.ibm.ims.dli.PSBFactory;
import com.ibm.ims.jdbc.IMSDataSource;
...
    IMSConnectionSpec connSpec
        = IMSConnectionSpecFactory.createIMSConnectionSpec();
    PSB psb;
    PCB pcb;

    connSpec.setDatastoreName(alias);
    connSpec.setDatabaseName(databaseName);
    // connSpec.setMetadataURL(url);---> deprecated

    connSpec.setUser(user);
    connSpec.setPassword(password);
    connSpec.setDriverType(driverType);

    if (driverType == IMSDataSource.DRIVER_TYPE_4) {
        connSpec.setSSLConnection(enableSSL);
        connSpec.setDatastoreServer(host);
        connSpec.setPortNumber(port);
    }

    try {
        psb = PSBFactory.createPSB(connSpec);
        pcb = psb.getPCB(pcbName);

        // do some work here

        psb.commit();// or alternatively, psb.rollback()

        psb.close();
        pcb.close();
    } catch (DLIException e) {
        // handle exception
    }
```

DL/I connection: For a DL/I client, the PCB is the target for other interactions.

8.7.2 Using the metadata information in the DL/I access

With traditional languages (COBOL, PL/I, C, and so on), reading a segment brings a complete record in to an I/O area known as a *byte area*. This area must be interpreted. The byte area contains value fields, with structures and substructures. In these languages, the extract of the elementary values, which can be in different formats (char, binary, or packed decimal), is straightforward. The extract is done by overlaying the I/O area with redefinitions (COBOL copybooks, and PL/I or C include members). This technique is not available for Java.

The changes to the DBDGEN macros are explained in 8.4.1, “DBD and PSB source changes” on page 294. The changes on DBDGEN, and the extensions to the Java access classes, allow for an enhanced access approach of the DL/I databases by Java programs. The changes in DBDGEN provide for the extension of the metadata in the DBD and to store it in the catalog. Other metadata information, implicitly present in the redefinition sections, can also be captured into the catalog.

Using the metadata, Java (JDBC and DL/I) offers powerful detailed and controlled access to the information in the databases. When you program a Java client to access the DL/I databases, you can use different styles:

- ▶ JDBC SQL approach, with or without IMS extensions
- ▶ IMS extended DL/I style

Struct and Array objects: Traditionally, when creating a Struct or an Array object, you must define it in a bottom-up manner that starts with the Struct attributes or the Array Elements.

IMS allows for the top-down creation of a Struct or an Array object. You can define the Struct or Array object and then set the attributes and elements. It avoids potential complexity when you deal with nested structures. It also allows for setting and getting Struct and Array attributes by the attribute name.

Along with the metadata support for Java, several enhancements are available for Java. The enhancements are described in the following sections.

Variable length segment support

The IMS Universal drivers for both type-2 and type-4 database access are enhanced to support variable length database segments. Support is added to the Universal drivers by the following maintenance:

- ▶ Version 11: APAR PM14766
- ▶ Version 12: APAR PM25951

Because the standards for JDBC do not have a concept of segment length, the default behavior is that the JDBC driver manages variable length segments internally. The behavior is managed for read, update, and insert operations, without any additional action from the client.

- ▶ SQL clients can directly read and update the length field of a variable length segment by explicitly requesting access when the connection is created.
- ▶ DL/I clients always have access to the length field.

Variable length segments contain a 2-byte length (LL) field that identifies the size of the segment instance. The Universal drivers are now sensitive to the LL field of a variable length segment. The drivers manage the I/O area of the segment instance on standard *create*, *retrieve*, *update*, and *delete* calls. A new Connection Property *llField* determines whether the LL field of the segment is displayed to the program or not.

The New Connection Property *llField* can be set to true/false:

- ▶ `llField = true`

The LL field of variable length segments is displayed to the user in both the SQL and DL/I interfaces. The user fully manages the LL field for the segment instance. The LL field is how users manage variable length segments.

- ▶ `llField = false` (default)

The IMS Universal driver automatically manages the LL field for the user.

Variable length support: Variable length support is introduced for both DLIModel and the IMS catalog metadata.

Example 8-21 shows the variable length information that related to the segment WARD in the DatabaseView.

Example 8-21 Variable segment length information in DatabaseView class

```
static DLISegment PHDAMVARWARDSegment = new DLISegment
    ("WARD","WARD",PHDAMVARWARDArray,32,900,DBType,PHAM,false);
//                                Minlength, Maxlength
```

Example 8-22 shows the variable length information in the catalog.

Example 8-22 Variable segment length information in IMS catalog: XML

```
<segment imsName="WARD" name="WARD">
  <phdam>
    <bytes minBytes="32" maxBytes="900"/> <=====minbytes,maxbytes
  </phdam>
  ....
</segment>
```

Structure support

Structures are data structures that store a combination of values. Structure support was added specifically for the IMS catalog in IMS V12 to represent common application metadata. Example 8-23 represents a simple structure with the name ADDRESS-INFO. In Java, it is an object.

Example 8-23 COBOL structure example

```
01 PERSON.
  02 PERSNR      PIC X(6).
  02 ADDRESS-INFO.
    04 CITY      PIC X(15).
    04 STREET    PIC X(25).
    04 ZIP       PIC X(5).
```

Structure metadata is stored in the IMS catalog. IMS Universal drivers can retrieve the metadata on Structs as XML through the GUR call, as shown in Example 8-24.

Example 8-24 Structure information in the IMS catalog

```

.....
<field name="ADDRESS_INFO">
  <startPos>6</startPos>
  <bytes>45</bytes>
  <marshaller encoding="CP1047">
    <typeConverter>STRUCT</typeConverter>
  </marshaller>
  <applicationDatatype datatype="STRUCT"/>
  <field imsDatatype="C" name="CITY">
    <startPos>1</startPos>
    <bytes>15</bytes>
    <marshaller encoding="CP1047">
      <typeConverter>CHAR</typeConverter>
    </marshaller>
    <applicationDatatype datatype="CHAR"/>
  </field>
  .....other fields
</field>

```

Example 8-25 shows the code for the JDBC SQL retrieve of the structure (PERSON_INFO). See the COBOL definition in Example 8-23 on page 335. In the code, you extract the subfield values from the structure object. Two approaches are shown in Example 8-25.

Example 8-25 JDBC SQL for structure retrieve

```

Connection conn = ....
Statement st = conn.createStatement("SELECT * FROM"
                                     + pcbName + ".PERSON WHERE PERSNR = 'XXXXXX?');
ResultSet rs = st.executeQuery();
rs.next();
// 2 ways to retrieve the information
//-----
// 1) standard SQL
Struct addressInfo = (Struct)rs.getObject("ADDRESS_INFO");
Object[] addressInfoAttributes = addressInfo.getAttributes();
String city = (String) (addressInfoAttributes[0]).trim();
String street = (String) (addressInfoAttributes[1]).trim();
String zip = (String) (addressInfoAttributes[2]).trim();
//-----
// 2) with IBM/IMS extensions
StructImpl addressInfoimpl = (StructImpl)rs.getObject("ADDRESS_INFO");
String city = addressInfoimpl.getString("CITY");
String street = addressInfoimpl.getString("STREET");
String zip = addressInfoimpl.getString("ZIP");
//-----

```

With the StructImpl class, you have direct access to the structure fields with the adequate getters.

Next, new values are provided for the fields and a JDBC/SQL update stores the new structure in the DL/I database, as shown in Example 8-26.

Example 8-26 Updating the PERSON segment with JDBC

```

Connection conn = ....
PreparedStatement ps = conn.prepareStatement("UPDATE " + pcbName + ".PERSON
      SET ADDRESS_INFO =? WHERE PERSNR = "XXXXXX");
//-----
// 1) standard SQL
Object[] newAddressInfoAttribute = new Object[]
      {"LEUVEN","BONDGENOTENLAAN", "3000"};
Struct newAddressInfoStruct = conn.createStruct(pcbName +
      ".PERSON.ADDRESS_INFO", newAddressInfoAttribute);
ps.setObject(1, newAddressInfoStruct);
//-----
// 2) with IBM/IMS extensions
StructImpl newaddressInfoImpl =
      (StructImpl)conn.createStruct(pcbName + ".PERSON.ADDRESS_INFO");
newaddressInfoImpl.setString("CITY","LEUVEN");
newaddressInfoImpl.setString("STREET","BONDGENOTENLAAN");
newaddressInfoImpl.setString("ZIP","3000");
ps.setObject(1, newaddressInfoimpl);
//-----
int numberOfSuccessfulUpdates = ps.executeUpdate();

```

The scenarios that are used in Example 8-26 showed code for a JDBC client. Example 8-27 shows a Java code excerpt for a retrieve with a DL/I client.

Example 8-27 DL/I access for structure retrieve

```

PCB pcb = ... {
    SSAList ssaList = pcb.getSSAList("PERSON");
//specify a qualified SSAList for segment PERSON = XXXXXX field PERSNR
    ssaList.addInitialQualification("PERSON","PERSNR",SSAList.EQUALS, "XXXXXX");
//Retrieve all fields from PERSON (fixed segment)
    ssaList.markAllFieldsForRetrieval("PERSON", true);
//Creates I/O area for data
    Path path = ssaList.getPathForRetrieve();
//retrieve the data
    pcb.getUnique(path, ssaList, true);
    DBStruct personinfo = (DBStruct)path.getObject("PERSON_INFO");
//-----
// 1)
Object[] addressInfoAttributes = personinfo.getAttributes();
String city = ((String)addressInfoAttributes[0]).trim();
String street = ((String) addressInfoAttributes[1]).trim();
String zip = ((String) addressInfoAttributes[2]).trim();
//-----
// 2)
String city = personinfo.getString("CITY").trim();
String street = personinfo.getString("STREET").trim();
String zip = personinfo.getString("ZIP").trim();

```

Arrays

If the correct metadata is assembled in the catalog, *arrays* are also supported. Arrays are data structures that store a repeating combination of values. Example 8-28 shows an array.

Example 8-28 COBOL array

```
01 STUDENT.  
  02 STUDENTNAME PIC X(25).  
  02 AGE PIC 9(2) COMP.  
  02 COURSE OCCURS 5 TIMES.  
    04 COURSENAME PIC X(15).  
    04 INSTRUCTOR PIC X(25).  
    04 COURSEID PIC X(5).
```

Dynamic and static array support was added specifically for the IMS catalog in IMS V12 to represent common application metadata. IMS Universal drivers support only static arrays. The drivers can retrieve the metadata on arrays as XML through the GUR call.

A proprietary method of setting the elements within a `DBArrayElementSet` object is added. The method is similar to a `ResultSet`. It is possible to position a specific element in an array with the following methods: `next()`, `previous()`, `first()`, `last()`, and `absolute(int index)`.

Also, common getters and setters are provided for nested fields within an array element:

- ▶ `setBoolean(String, Boolean)`
- ▶ `setString(String, String)`
- ▶ `getBoolean(String)`
- ▶ `getString(String)`

Example 8-29 shows the retrieve of the `STUDENT` segment, which contains an array. Notice the usage of the `DBArrayElementSet` class.

Example 8-29 Retrieve of the segment STUDENT

```
String[] coursename =new String[5];  
String[] instname =new String[5];  
String[] courseid =new String[5];  
st = conn.createStatement();  
rs = st.executeQuery("SELECT * FROM " + pcbName +  
                    ".STUDENT WHERE STUDENTNAME = XXXX" );  
  
rs.next()  
String studname = rs.getString("STUDENTNAME");  
short age = rs.getShort("AGE");  
// 2 ways to retrieve the information  
//-----  
// 1) standard SQL  
Array fivecourses = rs.getArray("COURSE");  
Struct[] fivecourseselem = (Struct[]) fivecourses.getArray();  
// Each array element is represented as a Struct  
for (int arrayIdx = 0; arrayIdx < fivecourseselem.length; arrayIdx++) {  
    Object[] courseinfo = fivecourseselem[arrayIdx].getAttributes();  
    coursename[arrayIdx] = (String) courseinfo[0];  
    instname[arrayIdx] = (String) courseinfo[1];  
    courseid[arrayIdx] = (String) courseinfo[2];  
}  
//-----  
// 2) with IBM/IMS extensions
```

```

int arrayIdx = 0;
ArrayImpl fivecoursesImpl = (ArrayImpl)rs.getArray("COURSE");
DBArrayElementSet fivecoursesArrayElementSet = fivecoursesImpl.getElements();
try {
    while (fivecoursesArrayElementSet.next()); {
        coursenam[arrayIdx] =
            fivecoursesArrayElementSet.getString("COURSENAME");
        String instname[arrayIdx] =
            fivecoursesArrayElementSet.getString("INSTRUCTOR");
        String courseid[arrayIdx] =
            fivecoursesArrayElementSet.getString("COURSEID");
        arrayIdx++;
    }
} catch (Exception e {
    .....
}
}
//-----

```

Example 8-30 shows the insertion of a new STUDENT segment. The two ways of JDBC coding are shown again.

Example 8-30 Insertion of a new STUDENT segment

```

PreparedStatement ps = conn.prepareStatement
    ("INSERT INTO " + pcbName + ".STUDENT (" + "STUDENTNAME", "AGE", "COURSE" +
    "VALUES( ? ? ? );");
String coursenm[] = {"HISTORY","ENGLISH","FRENCH","ITALIAN","JAVA"};
String teacher[] = {"SMITH","OBAMA","SARKOZY","PAOLO","SVLTEACHER"};
String courseid[] = {"10","51","52","55","40"};
ps.setString(1,"FREDERIK");
ps.setShort(2,41);
//-----
// 1) standard SQL
// fill now the ARRAY object
Struct[] courseArrayElements = new Struct[5];
for (int arrayIdx = 0; arrayIdx < 5; arrayIdx++) {
    Object[] courseAttributes = new Object[]
        {coursenm[arrayIdx], teacher[arrayIdx], courseid[arrayIdx]};
    courseArrayElements[arrayIdx] = conn.createStruct
        ("COURSE", courseAttributes);
}
Array courseArray = conn.createArrayOf("COURSE", courseArrayElements);
ps.setArray(3,courseArray);
//-----
// 2) with IBM/IMS extensions
int arrayIdx = 0;
ArrayImpl fivecoursesImpl = ((ArrayImpl) ((ConnectionImpl)conn).createArrayOf
    (pcbName + ".STUDENT.COURSE"));
DBArrayElementSet fivecoursesArrayElementSet = fivecoursesImpl.getElements();
try {
    while (fivecoursesArrayElementSet.next()); {
        fivecoursesArrayElementSet.setString("COURSENAME",coursenm[arrayIdx]);
        fivecoursesArrayElementSet.setString("INSTRUCTOR",teacher[arrayIdx]);
        fivecoursesArrayElementSet.setString("COURSEID",courseid[arrayIdx]);
        arrayIdx++;
    }
}

```

```

    } catch (Exception e) {
    }
    ps.setArray(3,fivecoursesImpl);
//-----
    int numberOfSuccessfulInserts = ps.executeUpdate();

```

Mapping support

A map is metadata that describes how a field (or set of fields) is mapped for a particular segment instance. Metadata captures the various cases, and for each case, defines the set of fields to be used for that case. Maps can be defined to the catalog.

Maps are interpreted at run time by the IMS Universal drivers and the applicable data elements are returned based on the runtime case of the segment instance. Figure 8-40 shows a case selection that is based on the control field Policy Type.

Policy Type	Property Type	Rooms	MValue	Address	Make	Model	Year	HValue	Color
M	-	-	-	-	Ford	Escort	1989	2K	Red
H	Single Family	5	500K	555 Disk Drive Way, 95141	-	-	-	-	-

Figure 8-40 Insurance segment mapped multiple ways based on the Policy Type control field

All case fields are displayed in the metadata. There is no concept of maps or cases at the SQL level. The support by metadata creates a requirement that all case fields have unique names in a table.

Figure 8-40 shows the SQLview of the data. This representation gives the impression that much space is empty on the disk. The reality is different. All maps must be the same length, and in this case, the physical view of the segment is similar to what is shown in Figure 8-41.

Policy Type	Make	Model	Year	HValue	Color
M	Ford	Escort	1989	2K	Red
	Property Type	Rooms	MValue	Address	
H	Single Family	5	500K	555 Disk Drive Way, 95141	

Figure 8-41 Segment view on disk

Figure 8-41 shows that the control field Policy Type is not really a part of the mapping.

Example 8-31 shows an excerpt of the case mapping information in the metadata.

Example 8-31 IMS Universal drivers pull metadata from the IMS catalog GUR call as XML

```

<field imsDatatype="C" imsName="POLTYPE" name="PolicyType"> <-CASE Control field
  <startPos>1</startPos>
  <bytes>1</bytes>
  <marshaller encoding="CP1047">

```

```

        <typeConverter>CHAR</typeConverter>
    </marshaller>
    <applicationDatatype datatype="CHAR"/>
</field>
<mapping dependingOnField="PolicyType">
    <case name="HOUSE"> .....<- CASE
        <dependingOnFieldValue valueDatatype="C" value="H"/>
        <field imsDatatype="C" imsName="PROPTYPE" name="PropertyType">
            <startPos>2</startPos>
            <bytes>15</bytes>
            <marshaller encoding="CP1047">
                <typeConverter>CHAR</typeConverter>
            </marshaller>
            <applicationDatatype datatype="CHAR"/>
        </field>
        ...
    </case>
    ...
</mapping>

```

Maps and cases are a new feature that are added to both the SQL and DL/I interfaces and are functionally identical. The use of case-mapping support offers the following specific create, retrieve, update, and delete behaviors:

- ▶ SQL Select and DL/I Read

When a case is not the active case that is based on the control field of the map, its fields are treated as null fields.

- ▶ SQL Insert and DL/I Create

You cannot insert values for the fields of a case unless the insert also includes the value for the control field that makes the case active.

- ▶ SQL Update and DL/I Update

You cannot update the values for the fields of a case unless the case is active, or you can update the values if the control field is also updated to a value that makes the case active.

- ▶ SQL Delete and DL/I Delete

No new behavior is observed.

Example 8-32 shows a case and mapping retrieve.

Example 8-32 Cases and mapping

```

st = conn.createStatement();
rs = st.executeQuery("SELECT * FROM " + pcbName + ".INSURANCES");
while (rs.next()) {
    byte policytype = rs.getBytes("PolicyType");
    switch (policytype) {
        case ('M'):
            String make = rs.getString("Make");
            String model1 = rs.getString("Model");
            short year = rs.getShort("Year");
            int mvalue = rs.getInt("MValue");
            String color = rs.getString("Color");
        case ('H'):
            String propertyType = rs.getString("Make");
    }
}

```

```

        short room = rs.getShort("Room");
        int hvalue = rs.getInt("HValue");
        String address = rs.getString("Address");
    }
}

```

Redefines

Redefines are overlapping fields. Redefines are a way of remapping a field in the database. See Example 8-33 for a COBOL structure example of a redefine.

Example 8-33 COBOL structure example

```

01 PERSON.
  02 ADDRESS PIC X(45).
  02 ADDRESS-INFO REDEFINES ADDRESS. <-----
    04 CITY PIC X(15).
    04 STREET PIC X(25).
    04 ZIP PIC X(5).

```

Example 8-34 shows the way this redefine information is stored in the catalog.

Example 8-34 Redefine metadata information in the IMS catalog

```

<field imsDatatype="C" name="ADDRESS">
  <startPos>1</startPos>
  <bytes>45</bytes>
  <marshaller encoding="CP1047">
    <typeConverter>CHAR</typeConverter>
  </marshaller>
  <applicationDatatype datatype="CHAR"/>
</field>

<field name="ADDRESS_INFO" redefines="ADDRESS" .....< redefine
  <startPos>1</startPos>
  <bytes>45</bytes>
  <marshaller encoding="CP1047">
    <typeConverter>STRUCT</typeConverter>
  </marshaller>
  <applicationDatatype datatype="STRUCT"/>
  ...
</field>

```

Fields that overlap can be interchangeable. The search performance of a query depends on the type of field in the qualification statement:

- ▶ Key fields: Fastest
- ▶ Searchable fields
- ▶ Not searchable (that is, application defined fields)

The universal drivers promote a field upward when issuing queries against IMS.

The following example shows how the key field, KEY, and the non-key field, NONKEY, redefine each other:

```

SELECT * FROM TBL WHERE NONKEY=A ==>becomes
SELECT * FROM TBL WHERE KEY=A

```


The trace and log files show the promotion in the SSA list that is sent from the IMS Universal driver to IMS.

Public converter interfaces

Public interfaces are added to the internal type converters of the IMS Universal drivers for clients to use in implementing their own type converter routines. Support is added in IMS V10 and later.

The new ConverterFactory class allows users to create the following converter interfaces:

- ▶ DoubleConverter
- ▶ FloatConverter
- ▶ IntegerConverter
- ▶ LongConverter
- ▶ PackedDecimalConverter
- ▶ ShortConverter
- ▶ StringConverter
- ▶ UByteConverter
- ▶ UIntegerConverter
- ▶ ULongConverter
- ▶ UShortConverter
- ▶ ZonedDecimalConverter

The converter classes contain a *getter and setter method* for converting the data type to a binary representation.

The IMS catalog is built to store information about the fields with a user-defined type. This defined type allows users to interpret binary data that is stored in IMS in whatever form is required for their application.

Example 8-35 shows the IMS catalog metadata that contains a user-defined type.

Example 8-35 Field with a fully defined converter class

```
<field name="PACKEDDATEFIELD">
  <startPos>40</startPos>
  <bytes>5</bytes>
  <marshaller encoding="">
    <userTypeConverter>class://com.ims.PackedDateConverter</userTypeConverter>
    <property name="pattern" value="yyyyMMdd"/>
    <property name="isSigned" value="N"/>
  </marshaller>
  <applicationDatatype datatype="OTHER"/> <===== set to Other =====>
</field>
```

User Type converters must extend the com.ibm.ims.dli.types.BaseTypeConverter abstract class.

Universal drivers include a user-defined type converter with its source for the PackedDate type, which stores date information as a PackedDecimal field, as shown in Example 8-36.

Example 8-36 User-defined type converter for the PackedDate type

```
public class PackedDateConverter extends BaseTypeConverter {

    public Object readObject(byte[] ioArea,int start,int length,Class objectType,
        Collection<String> warningStrings) throws ConversionException {
```

```

    ...
}

public void writeObject(byte[] ioArea, int start, int length, Object object,
    Collection<String> warningStrings) throws ConversionException {
    ...
}
}

```

Application-transparent metadata access

Users can explicitly issue a GUR call through the following method, which is only available for DL/I clients:

```
PCB.getCatalogMetadataAsXML(String resourceName, byte[] resourceType)
```

Here are the input parameters for this method:

- ▶ **resourceName**
The name of the PSB or DBD resource to be retrieved.
- ▶ **resourceType** in the format PCB.PSB_RESOURCE or PCB.DBD_RESOURCE
Identifies whether a PSB or a DBD resource is retrieved.

The call returns an XML document that contains the metadata for the resources that are requested. The XML document conforms to the IMS managed schemas. The following schemas are used when providing XML information from the IMS catalog:

- ▶ DFS3XDBD.xsd
- ▶ DFS3XPSB.xsd

The source of the XML Schema Definitions (XSDs) is provided in the IMS sample library (SDFSSMPL).

Example 8-37 shows the explicit GUR call.

Example 8-37 Example of an explicit GUR call

```
byte[] gurOutput = pcb.getCatalogMetadataAsXML("STLIVP1", PCB.PSB_RESOURCE);
```

Example 8-38 shows the `getCatalogMetaDataAsXML` public method definition.

Example 8-38 Example of a `getCatalogMetaDataAsXML` public method definition

```

/**
 * This method returns a byte array containing the requested catalog resource
 * as an XML document.
 * <p>The following code fragment illustrates how to retrieve the timestamp
 * (TSVERS) value from the IMS Catalog.
 * <blockquote>
 * <pre>
 * PCB pcb = pcb.getPCB("DFSCAT00");
 * SSAList ssaList = pcb.getSSAList("HEADER", "DBD");
 * Path path = ssaList.getPathForRetrieveReplace();
 * pcb.getUnique(path, ssaList, false);
 * String timestamp = path.getString("TSVERS");
 * </pre>
 * </blockquote>

```

```

*
* @param resourceName the name of the PSB or DBD resource in the catalog
* @param resourceType the type of resource (PCB.PSB_RESOURCE or
PCB.DBD_RESOURCE)
* @param timestamp the TSVERS version for the resource following the
*       pattern yyDDHHmmssff
* @return the resource metadata in XML
* @throws DLIException if the resource was not found in the catalog or an
error
*       occurs during processing
* @see #PSB_RESOURCE
* @see #DBD_RESOURCE
*/
public byte[] getCatalogMetaDataAsXML(String resourceName,
                                     byte[] resourceType, String timestamp)
    throws DLIException
;

```



IMS Explorer and Open Database connectivity

This chapter describes how the IBM IMS Enterprise Suite Explorer for Developers simplifies the development of IMS applications.

IMS Explorer combines a graphical front end with an integrated development environment (IDE) that is built on the Eclipse platform, which is an open software development environment that provides a standard GUI-based framework. As a result, IMS Explorer can be integrated with other Eclipse-based plug-in tools, such as IBM Rational Developer for System z and IBM Data Studio, to support the entire IMS application development cycle. Moreover, developers familiar with Eclipse-based tools can use their existing skills to immediately start manipulating IMS data through IMS Explorer.

This chapter covers the following topics:

- ▶ IMS Explorer for Developers
- ▶ How IMS Enterprise Suite extends access to IMS

9.1 IMS Explorer for Developers

IMS Explorer for Developers is more than a GUI; it is an easier way to access IMS PSBs and DBDs. This section explores the challenges around application development for IMS and how IMS Explorer addresses some of these challenges and how IMS Explorer is part of the wider IMS simplification effort.

Here are some challenges that IMS customers have:

- ▶ There is a shrinking knowledge base around IMS and the hierarchical database model.
- ▶ It is difficult to find DLI programmers.
- ▶ There are fewer experienced COBOL and PL/I programmers.
- ▶ There is a lack of integrated development solutions and tools.
- ▶ It is difficult to test and deploy applications.

Figure 9-1 shows IMS as it exists today with IMS TM and DB.

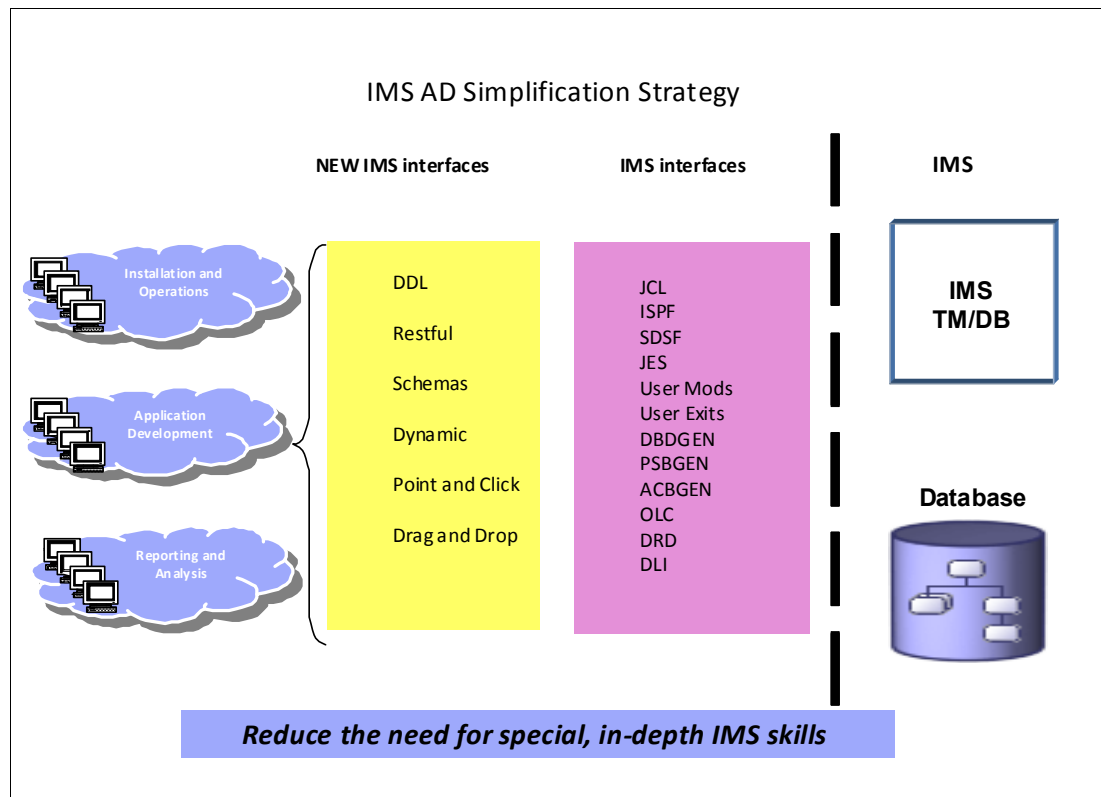


Figure 9-1 Simplification strategy

The box in purple shows some of the IMS interfaces that are used today. These are interfaces for systems programmers, DBAs, and application programmers. These interfaces will continue to be supported in the future.

The box in yellow shows some of the new interfaces that simplify the way people deal with IMS. It is a move toward more standard interfaces that do not require IMS, Assembler, or mainframe specific knowledge.

The vertical black dashed line represents the external interfaces, which move farther away from IMS as the additional abstraction layer of interfaces is added.

IMS Explorer addresses some of the application developer (AD) challenges by providing an easy-to-use interface to simplify common IMS application development tasks, and by being able to integrate with other IBM solutions.

The overall goal of IMS Explorer is to complement the IBM portfolio of Enterprise Modernization products on the Eclipse platform to help users complete their end-to-end IMS application development task flow without switching platform (for example, without switching to TSO). Particularly, IMS Explorer aims to explore and fill gaps in the end-to-end IMS AD task flow by providing select targeted functions, such as database and program visualization and definitions.

IMS Explorer can be integrated with other IBM solutions that are also based on Eclipse (such as Rational Developer for System z / Rational Application Developer, IBM Optim™ Data Studio, and IBM CICS Explorer®). IMS Explorer is one piece of the puzzle that, when it is integrated with other IBM products, provides an end-to end solution for IMS application development simplification.

If you are familiar with the DLIModel utility, you can think of the IMS Explorer as an enhanced version of that tool. Existing DLIModel utility users can start using the IMS Explorer immediately.

For host connectivity, the IMS Explorer requires the user to provide the DBD and PSB source so that they can be imported into the IMS Explorer, and then the user can then do visualization, editing, and so on. The goal is to add FTP capabilities to the IMS Explorer so that the user can get the required source files directly from a PDS without having to manually download them to the workstation using some other FTP process.

COBOL/PLI copybook importers that are in the DLIModel Utility are added in to the IMS Explorer, which allows the user to add a copybook definition in to a DBD segment and provide the following features:

- ▶ Easier visualization and editing of IMS Database and Program (PSB) definitions through the following functions:
 - Graphical editors can perform the following tasks:
 - Display IMS database hierarchical structures.
 - Display, create, and edit PSBs.
 - Change and add fields on a DBD.
 - You can import COBOL copybooks and PL/I data structures to a database segment.
 - You can generate DBD and PSB sources.
 - You can generate the metadata that is needed by Java applications and for SQL access from IMS Explorer.
- ▶ Ability to easily access IMS data by using SQL statements, which allows the following functions
 - You can use IMS V11 Universal JDBC driver and Open Database Connectivity with a z/OS system.
 - You can browse a data set and submit JCL.
 - You can import and export DBD and PSB source files from a data set to the IMS Explorer and vice versa.

Currently, the way to read and understand an IMS DB is by reading its source, as shown in Figure 9-2, where the source is in text. This task is easy for an experienced DBA, but not for someone who is new to IMS.

DATASET DD1=DFSDLR	00350001
SEGM NAME=DEALER,PARENT=0,BYTES=61	00360001
FIELD NAME=(DLRNO,SEQ,U),BYTES=4,START=1,TYPE=C	00370001
FIELD NAME=DLRNAME,BYTES=30,START=5,TYPE=C	SECINDX1 SEARCH1 00380001
FIELD NAME=CITY,BYTES=10,START=35,TYPE=C	SECINDX1 SEARCH2 00390001
FIELD NAME=ZIP,BYTES=10,START=45,TYPE=C	SECINDX1 SUBSEQ 00400001
FIELD NAME=PHONE,BYTES=7,START=55,TYPE=C	SECINDX1 DUPD 00410001
LCHILD NAME=(SINDEXB,SINDEX22),POINTER=INDX	00420001
XDFLD NAME=XFLD2,SEGMENT=MODEL,	X00430001
SRCH=(MAKE,MODEL),	X00440001
SUBSEQ=(YEAR,/SX1),	X00450001
DDATA=COUNT	00460001
SEGM NAME=MODEL,PARENT=DEALER,BYTES=37	00470001
FIELD NAME=(MODKEY,SEQ,U),BYTES=24,START=3,	X00480001
TYPE=C	SECINDX2 SEARCH 00490001
FIELD NAME=MODTYPE,BYTES=2,START=1,TYPE=C	00500001
FIELD NAME=MAKE,BYTES=10,START=3,TYPE=C	SECINDX2 SEARCH 00510001
FIELD NAME=MODEL,BYTES=10,START=13,TYPE=C	SECINDX2 SEARCH 00520001

Figure 9-2 SDFSISRC(DFS AUTDB)

9.2 How IMS Enterprise Suite extends access to IMS

This section shows how IMS Explorer for Development displays a DBD, starting with the window shown in Figure 9-3.

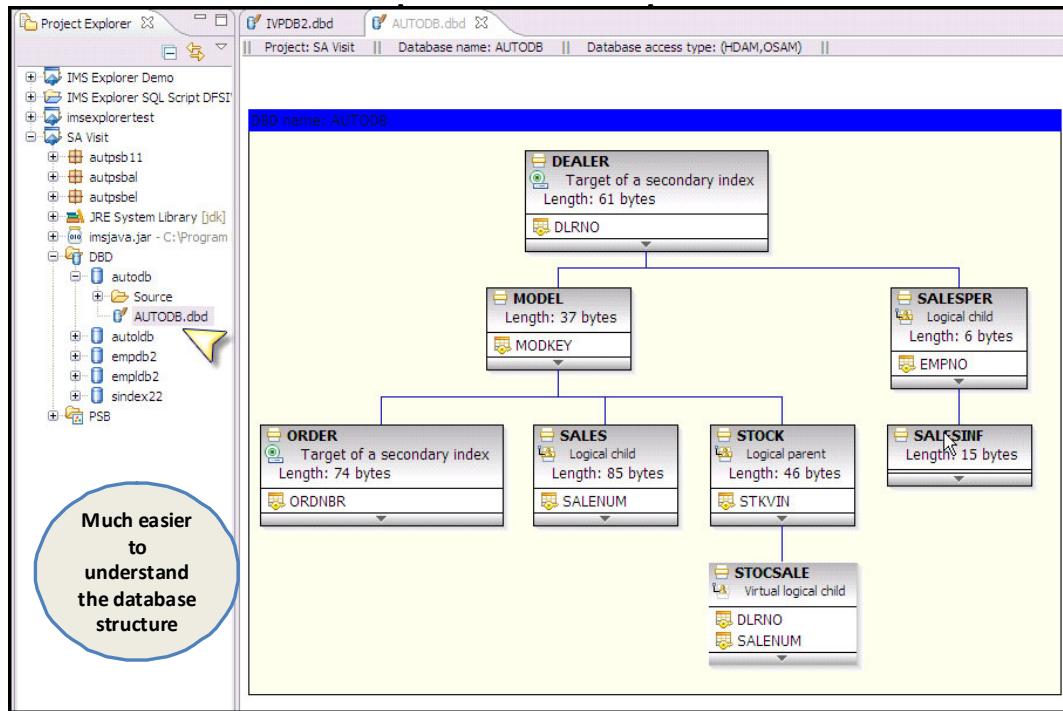


Figure 9-3 AUTODB DBD Source

Note: IMS Explorer is part of the IMS Enterprise Suite V22. You can download it at no charge from the following website:

<http://www-01.ibm.com/software/data/ims/>

Figure 9-4 shows a DBD source that uses the IMS Explorer DBD Graphical Editor. The dashed lines show the logical relationships between the related DBD sources.

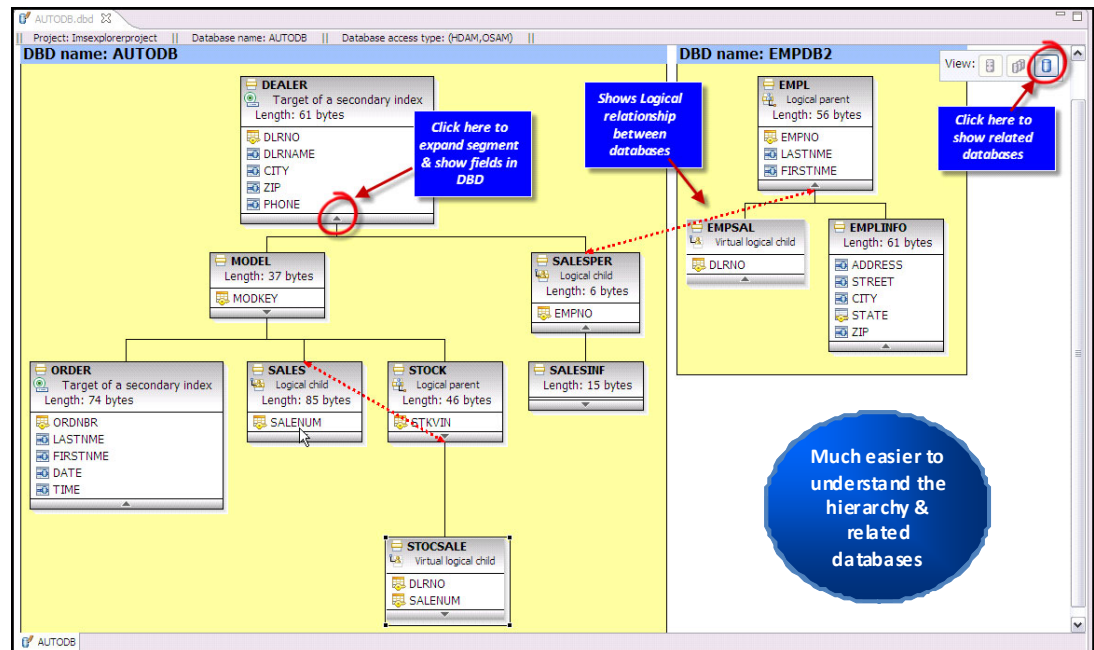


Figure 9-4 Showing the logical relationship

When a user clicks the dashed line of a logical relationship, the type of relationship is displayed in the Properties view (for example, bidirectional virtually paired) and the Parent Segment and the Child Segment.

In addition, when a user imports a DBD that contains logical relationships, they see all the other DBD sources that are needed to see all the relationships.

If you click the dotted line that is shown in Figure 9-5, the window shows the type of relationship in the Properties view. To print the layout, right-click the diagram and select **Print**.

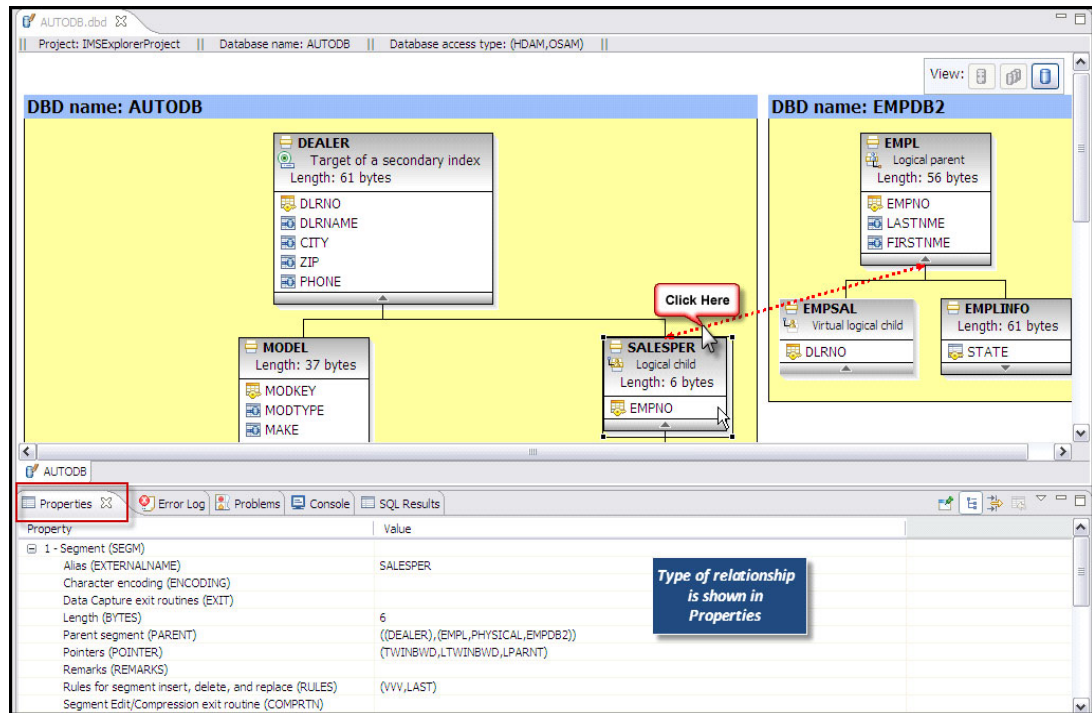


Figure 9-5 Type of relationships in the Properties view

Today, PSBs and PCBs are defined using macros that are assembled through a PSBGEN. Figure 9-6 shows a typical PCB definition.

```

AUTS2PCB PCB TYPE=DB,DBDNAME=AUTOLDB,PROCOPT=GRP,KEYLEN=64,
          PROCSEQ=INDEX22
          SENSEG NAME=DEALER,PARENT=0
          SENSEG NAME=MODEL,PARENT=DEALER
          SENSEG NAME=STOCK,PARENT=MODEL
*
AUSI2PCB PCB TYPE=DB,DBDNAME=INDEX22,PROCOPT=GRDP,KEYLEN=28
          SENSEG NAME=SINDXB,PARENT=0

EMPLPCB PCB TYPE=DB,DBDNAME=EMPLDB2,PROCOPT=AP,KEYLEN=10
          SENSEG NAME=EMPL,PARENT=0
          SENSEG NAME=DEALER,PARENT=EMPL
          SENSEG NAME=SALESINF,PARENT=DEALER
          SENFLD NAME=QUOTA,START=1
          SENFLD NAME=SALESYTD,START=7
          SENSEG NAME=EMPLINFO,PARENT=EMPL
          PSBGEN PSBNAME=AUTPSB11,LANG=JAVA
          END
  
```

Figure 9-6 Typical PCB definition within a PSB

With IMS Explorer for Development, you can display and build the PSB and all the PCBs in a graphic form, as shown in Figure 9-5 on page 353. In this figure, IMS Explorer generates the PSB source.

You can use the IMS PCB editor to define which segments an application program is sensitive to in the database program communication block (PCB) for the hierarchy that contains those segments. An IMS application program can access only data to which it is sensitive. Segment sensitivity can prevent an application program from accessing all the segments in a particular hierarchy. The sensitivity tells IMS which segments in a hierarchy that the program is allowed to access.

To specify the segment sensitivity, open the IMS PCB editor, as shown in Figure 9-7, on the PSB that contains the full-function or Fast Path database PCB and complete the following steps:

1. Select the database PCB statement.
2. Open the PCB editor by clicking **Edit PCB Segment**.
3. To mark a database PCB segment as sensitive, select the check box. Clear the check box to mark the database PCB segment as insensitive.
4. Click **File** → **Save**.

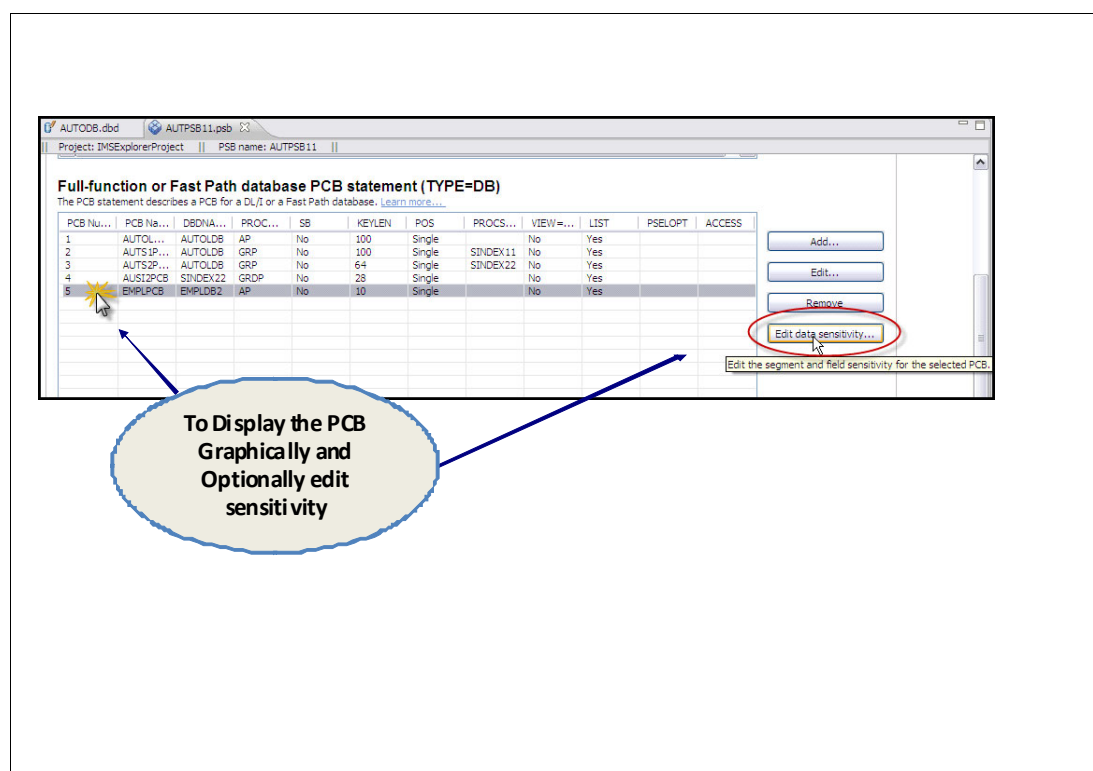


Figure 9-7 Display and build the PSB and all the PCBs

Use the **SENSEG** statement with the database PCB statement to define a hierarchically related set of segments that an application program is sensitive to. This segment set can physically exist in one database or can be derived from several physical databases.

Figure 9-8 shows an update of the PHONEBOOK DataBase by using the Data Source Explorer. Open the DFSIVP37 PSB, right-click, and select **PHONEBOOK** → **Data** → **Edit**, which opens the table where you can add or delete rows. Click **Save**.

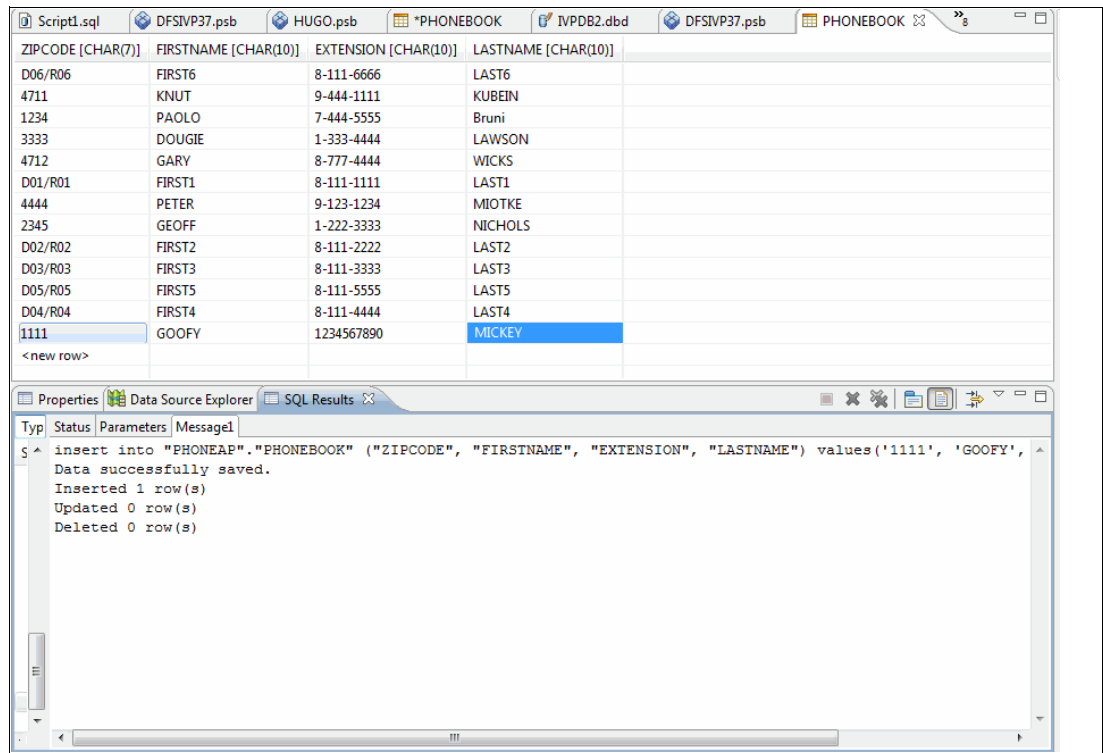


Figure 9-8 Table update of the Phonebook DB

As an example, add a few rows and view the SQL Result property to see how the SQL syntax looks, as shown in Example 9-1.

Example 9-1 SQL insert sample

```

insert into "PHONEAP"."PHONEBOOK" ("ZIPCODE", "FIRSTNAME", "EXTENSION",
"LASTNAME") values('1111', 'GOOFY', '1234567890', 'MICKEY')

```

If you clear **Disable Trace** in the Trace tap of the Driver Properties, you see the data flow shown in Figure 9-9.

```

FINER: ENTRY segmentName: PHONEBOOK fieldName: FIRSTNAME field offset: 10 field length: 10
value: GOOFY
Aug 01, 2013 2:05:06 PM com.ibm.ims.dli.PathImpl setValue(String, FieldEntry, String, Object)
FINER: RETURN
Aug 01, 2013 2:05:06 PM com.ibm.ims.dli.PathImpl setValue(String, FieldEntry, String, Object)
FINER: ENTRY segmentName: PHONEBOOK fieldName: EXTENSION field offset: 20 field length: 10
value: 1234567890
Aug 01, 2013 2:05:06 PM com.ibm.ims.dli.PathImpl setValue(String, FieldEntry, String, Object)
FINER: RETURN
Aug 01, 2013 2:05:06 PM com.ibm.ims.dli.PathImpl setValue(String, FieldEntry, String, Object)
FINER: ENTRY segmentName: PHONEBOOK fieldName: LASTNAME field offset: 0 field length: 10
value: MICKEY
Aug 01, 2013 2:05:06 PM com.ibm.ims.dli.PathImpl setValue(String, FieldEntry, String, Object)
FINER: RETURN
Aug 01, 2013 2:05:06 PM com.ibm.ims.drda.db.T4PCBImpl performMapValidationForInsert(Path)
FINER: ENTRY path: [ibm][ims][common][common][thread:1][tracepoint:1][PathImpl.toString]
[ibm][ims][common]          DISPLAY:          (ASCII)          (EBCDIC)
[ibm][ims][common]          0 1 2 3 4 5 6 7    8 9 A B C D E F    0123456789ABCDEF
0123456789ABCDEF
[ibm][ims][common] 0000    D4C9C3D2C5E84040    4040C7D6D6C6E840    .....@@@.....@    MICKEY
GOOFY
[ibm][ims][common] 0010    40404040F1F2F3F4    F5F6F7F8F9F0F1F1    @@@@.....
123456789011
[ibm][ims][common] 0020    F1F1404040000000    ..@@@...          11    ...

```

Figure 9-9 JDBC Trace

9.2.1 IMS Explorer demonstration

There are two IMS Explorer Versions that are available: a stand-alone version and a shell version for Rational Developer for System z or Data Studio. In this example, we use the shell version on Rational Developer for System z.

To start IMS Explorer, in the Windows main menu, click **Start → All Programs → IBM Software Delivery Platform → IMS Enterprise Suite Explorer for Development**. Specify the workspace to be used or use the default. A “Welcome Page” with an extensive Help menu might open.

In our example, we create an IMS Explorer Project by clicking **File → New → other → IMS → IMS Explorer Project**.

The Project wizard opens and prompts you for the Project Name. In our example, we choose **ITSO Redbooks Explorer Project** and click **Next**. The Select Import Source window, which is shown in Figure 9-10 on page 357, opens, and you can choose where you want your source imported from. In our example, we select **IMS catalog**, which has been a new feature since IMS V12 and later.

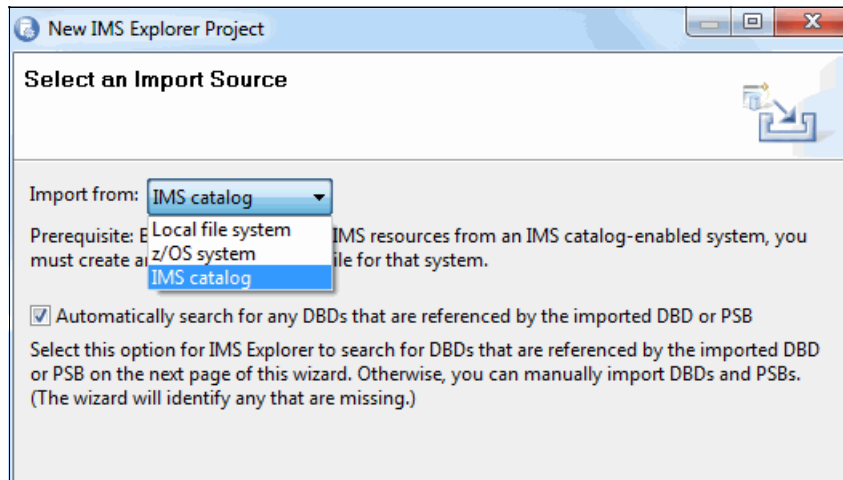


Figure 9-10 Select Import Source

You are prompted to choose a connection profile from a drop-down menu, as shown in Figure 9-11. If no connection profile exists, the window that is shown in Figure 9-14 on page 361 opens, and you would follow the steps in 9.2.4, “Establishing the host connection with IMS Explorer for Developer” on page 361. In our example, there is already a connection that is established.

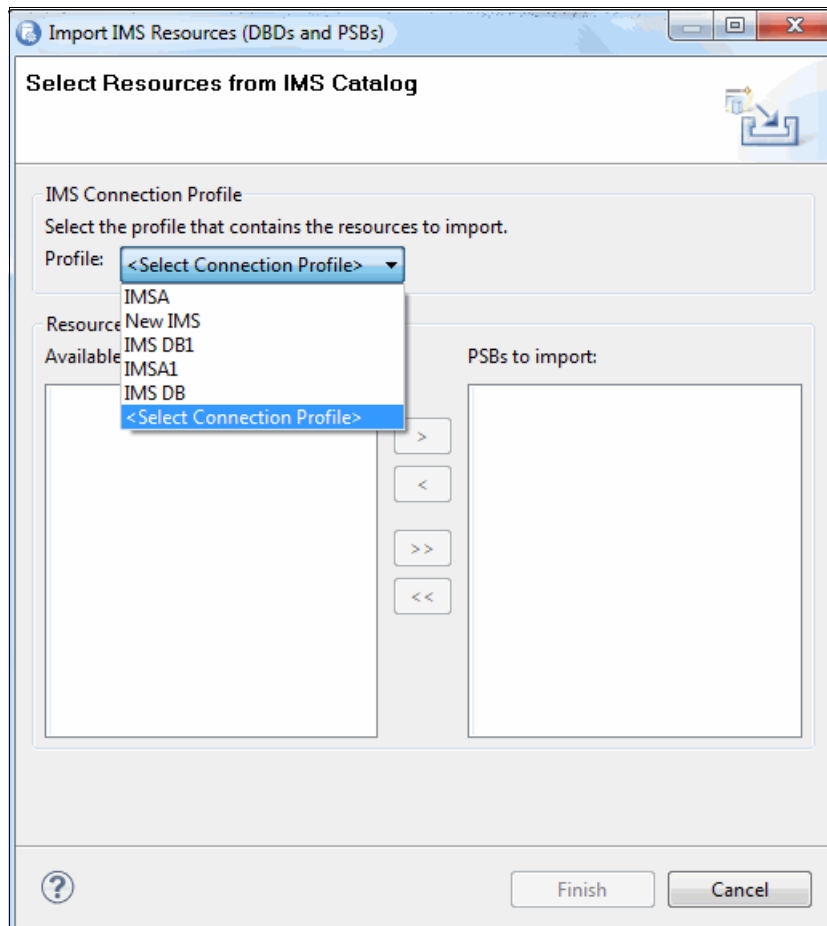


Figure 9-11 Select a connection profile

In this example, choose the host connection and get the resources that are available from the catalog. Choose DFSIVP37, which is the Phonebook application, to import, as shown in Figure 9-12.

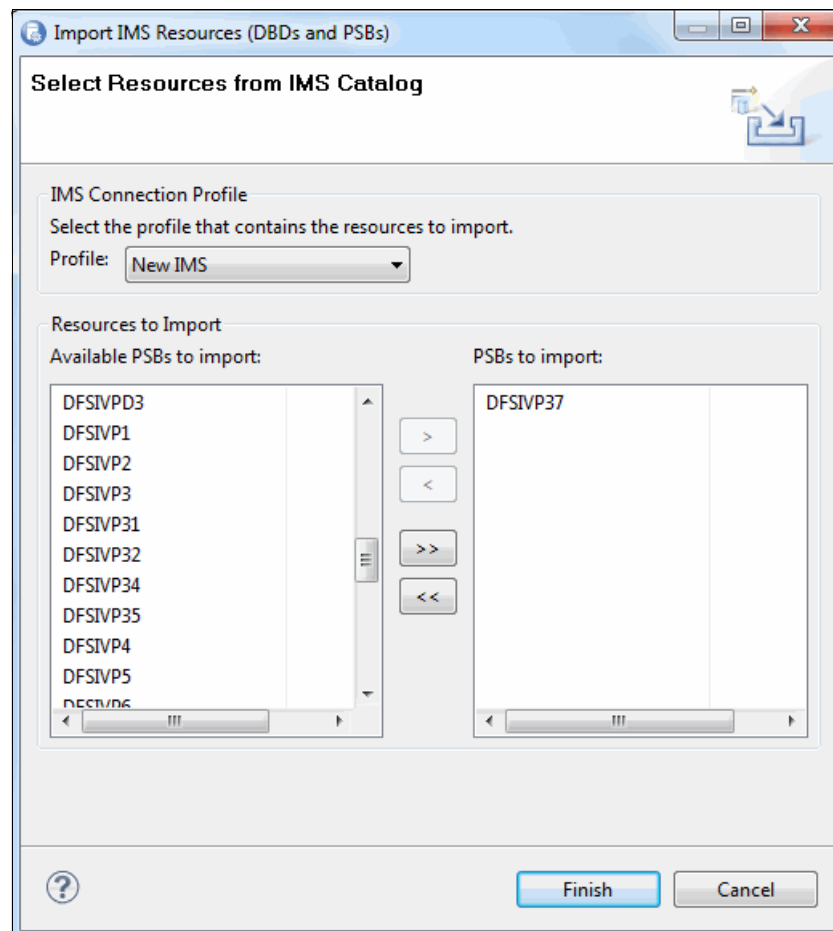


Figure 9-12 Available PSBs to import

Now you can open PSB or DBD Editors to modify the SENSEG statement or insert or modify data, as shown in Figure 9-13 on page 359.

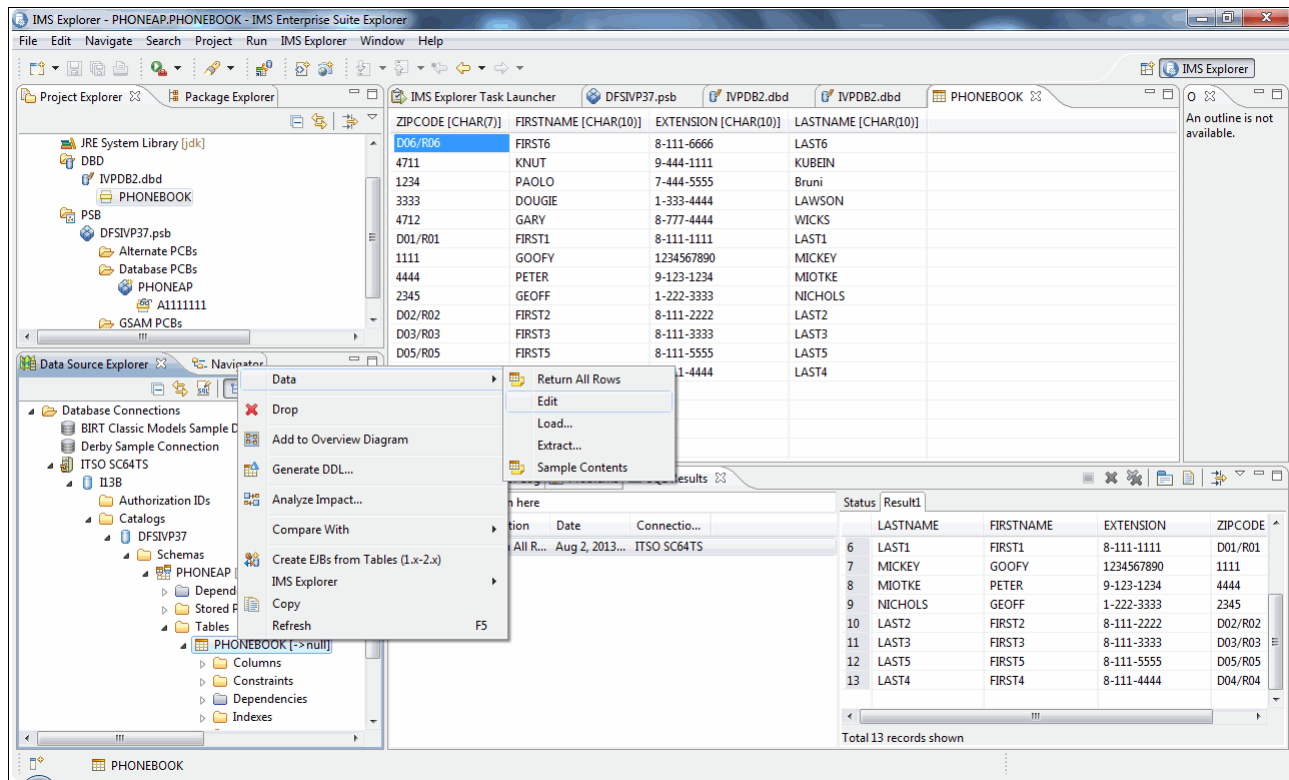


Figure 9-13 Modify or add data

In the Data Source Explorer view, click **Tables** → **PHONEBOOK** and right-click **Data Edit**. The columns and rows display. You can add, modify, and delete the data. After you make your changes, click **Save**.

9.2.2 Creating or modifying full-function or Fast Path database PCBs

You can use the full-function or Fast Path database PCB statement section of the PSB summary page to create or modify a full-function or Fast Path database PCB statement.

Open the PSB summary page for the PSB that contains the full-function or Fast Path database PCB and complete the following steps:

1. From the full-function or Fast Path database PCB statement section of the PSB summary page, you can create a full-function or Fast Path database PCB statement, and remove and modify existing statements.
 - To create a PCB statement, click **Add**. A new row is added to the table and the statement properties you can modify are displayed. A sequential PCB number uniquely identifies the new row.
 - To remove a PCB statement, click a row in the table to select it, and then click **Remove**. To remove multiple statements, select the first row that you want to remove, press and hold the Shift key, and click the last row that you want to remove, and then click **Remove**.
 - To edit a PCB statement, click a row in the table to select it, then click **Edit** and modify the properties. You can edit only one statement at a time.

You can specify values for these statement properties. These values are described in Table 9-1.

Table 9-1 Statement properties

Property	Required or optional	Description
PCB name	Required if LIST=N0 or if you want to use a Java metadata class	A 1 - 8 alphanumeric character PCB name or PCB label that must be unique within the PSB. The name cannot start with a numeric character.
Database name	Required	A 1 - 8 alphanumeric character name of the physical or logical database description (DBD) to be used as the primary source of database segments for this PCB.
Processing options (PROCOPT)	Required	The processing options on sensitive segments that are declared in this PCB. You can specify up to four options. The default is A.
Use sequential buffering (SB)	Optional	Specifies whether this PCB uses sequential buffering. The default is no.
Longest concatenated key (KEYLEN)	Required	Length of the longest concatenated key for a hierarchic path of sensitive segments that the application program uses in the logical data structure.
Positioning for the logical data structure (POS)	Optional	Single or multiple positioning provides a functional variation in the call. The default is Single.
Secondary processing sequence index name (PROCSEQ)	Optional	The name of a secondary index that is used to process the database that is named in the DBDNAME parameter through a secondary processing sequence. This property is valid only if a secondary index exists for this database.
Use MSDB commit view (VIEW=MSDB)	Optional if the PSB refers to a DEDB database	Specifies the main storage database commit view (Yes) or the DEDB commit view (No) for DEDBs. The default is No.
Include named PCB in the PCB list (LIST)	Optional	Specifies whether to include this PCB in the PCB list that is passed to the application program at entry. The default is Yes.

- Optional: You can open the database PCB editor to define which segments and fields in the database PCB that an IMS application program is sensitive to. To open the editor, click **Edit PCB Segment**.
- Click **File** → **Save**.

9.2.3 Connecting to an IMS database for SQL access

You can create and manage connections to IMS databases by using the Data Source Explorer. Use the New Connection wizard to create a connection profile so that you can connect to an IMS database and browse data objects.

Make sure that the host system that you are connecting to is set up to use Open Database. For more information, see the “Configuring IMS support for the IMS Universal drivers (System Definition)” topic at the following website:

http://pic.dhe.ibm.com/infocenter/dzichelp/v2r2/index.jsp?topic=%2Fcom.ibm.ims12.doc.sdg%2Fims_opendatabasesetupoverview.htm

To connect to an IMS database, complete the following steps:

1. In the Data Source Explorer, right-click the **Database Connections** folder and click **New**.
2. Follows the procedure that is described in 9.2.4, “Establishing the host connection with IMS Explorer for Developer” on page 361
3. Optional: On the Tracing tab, clear the **Disable tracing** check box to turn on tracing. Turning on tracing creates a log file of your activity that helps with troubleshooting problems.
4. Optional: On the Optional tab, specify whether to use an SSL for your connection, a data store, a log-in timeout value, and any additional property values you want to add to the JDBC connection URL.
5. Click **Finish**.

The connection is displayed in the Data Source Explorer.

9.2.4 Establishing the host connection with IMS Explorer for Developer

If you must establish a host connection because the window shown in Figure 9-14 opens, go to the Data Source Explorer, right-click, and select **Database Connections** → **New**. The New Connection wizard window opens shows up, as shown in Figure 9-15 on page 362. Select **IMS**.

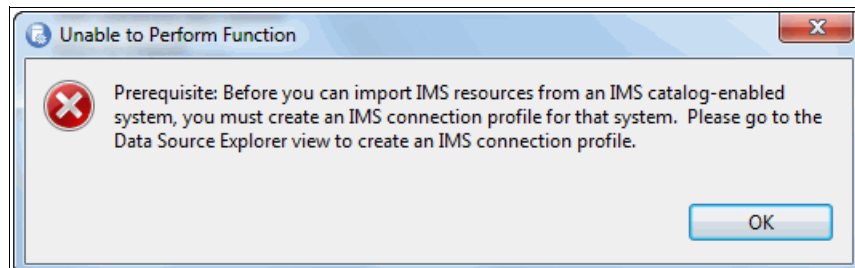


Figure 9-14 Unable to Perform Function

Set all the connection parameters in the General tab of the Properties field, as shown in Figure 9-15:

- ▶ Connection name: Set to the data store name if you are using the ODBM alias and an alias is needed. In this example, use I13B for the connection name.
- ▶ Host: Enter an IP address or host name.
- ▶ Port number: Use the ODBM port. Run the **VIEWHWS** command to find the port number with a D at the end of the port number (in this example, 6200D).
- ▶ Metadata: Set to Catalog for this example.
- ▶ PSB: Set to DFSIVP37 for this example.

In the Tracing tab, clear **Disable tracing**, and choose all the levels, and then click **Test Connection**. The test should succeed.

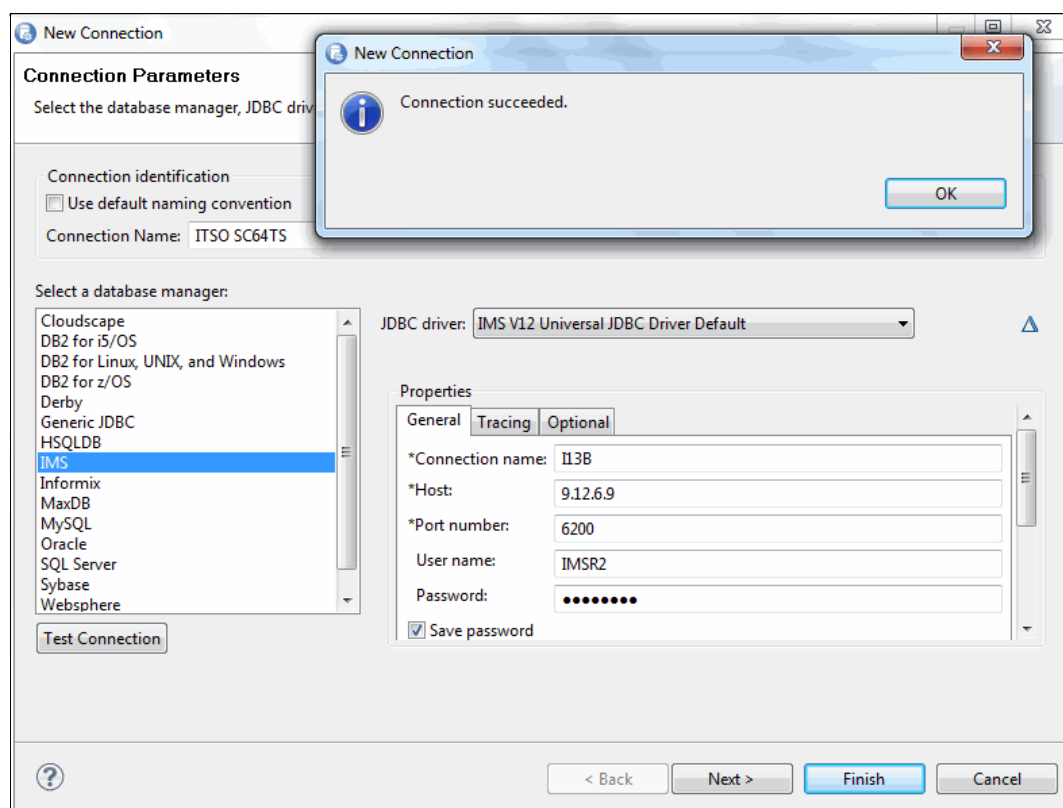


Figure 9-15 Connection Parameters

Make sure that you have the correct JDBC driver selected; in this example, use the IMS V12 JDBC Universal driver. Click the three handles beside the drive to add a driver. In this example, add the IMS V13 driver. The JDBC Driver JAR file can be found on your IMS host Environment by using the OMVS shell in TSO, but an easier way is to use Rational Developer for System z and open your z/OS UNIX files. In this example, open **root** → **usr** → **lpp** → **ims13q** or **imsjava**, as shown in Figure 9-16 on page 363. Here you find the **imsudb.jar** driver, which was copied in to a local folder for easy reference.

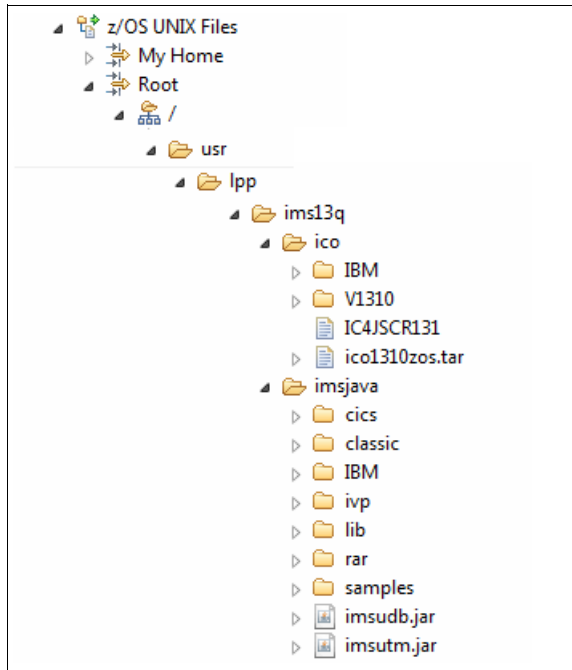


Figure 9-16 UNIX file view by using Rational Developer for System z

9.2.5 Creating and running SQL queries against an IMS database

You can access and manipulate data that is stored in an IMS database by creating and running SQL queries in IMS Explorer.

To create and run custom SQL queries against the IMS database, complete the following steps:

1. Create a data development project. The data development project is required to create and run custom SQL scripts.
 - a. From the main menu, click **File** → **New** → **Data Development Project**.
 - b. Enter a name for the data development project.
 - c. Click **Next**.
 - d. Select the connection name of the IMS database.
 - e. Click **Finish**.
2. Create an SQL script by using the SQL Query Builder. In the SQL script, specify the SQL query statements to run against the IMS database.
 - a. In the Data Project Explorer view, expand the data development project that you created in step 1.
 - b. Create an SQL script. Right-click the SQL Scripts folder and select **New** → **SQL** or **XQuery Script**.
 - c. Enter a name for the SQL script.
 - d. Use one of the following options to create an SQL script:
 - i. To enter the SQL statement as text, select **SQL and XQuery editor**.
 - ii. To graphically create the SQL statement, select **SQL Query Builder**.

Tip: More graphical semantic help is provided if you use the query builder, as shown in Figure 9-17.

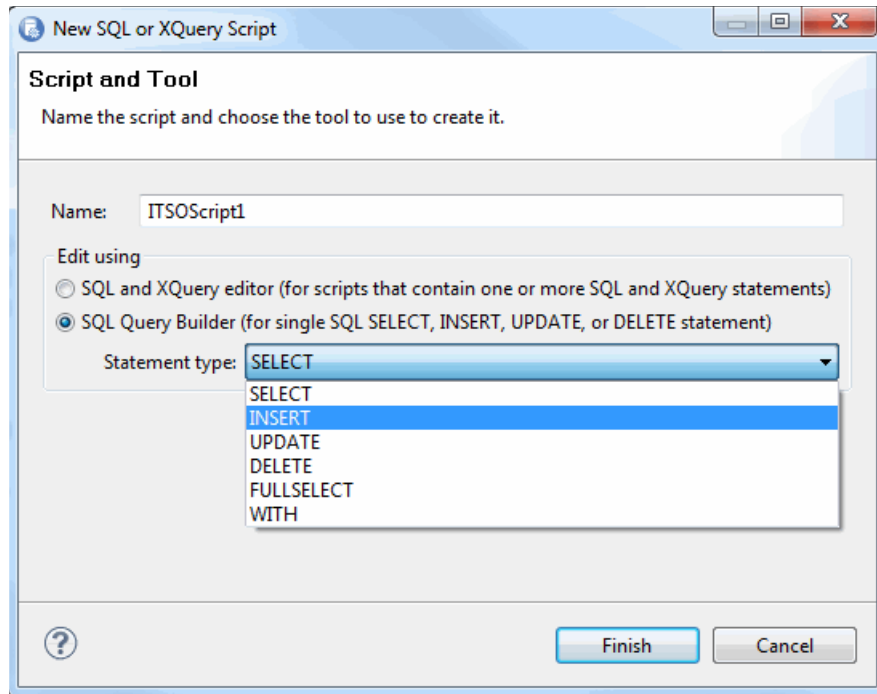


Figure 9-17 SQL Query Builder

- e. Click **Finish**.
3. To view the results in the SQL Results view, right-click the script that you created in step 2 on page 363. Select **Run SQL** to run the script.

Figure 9-18 shows the query result.

Status	Result1				
	LASTNAME	FIRSTNAME	ZIPCODE	EXTENSION	
1	LAST6	FIRST6	D06/R06	8-111-6666	
2	KUBEIN	KNUT	4711	9-444-1111	
3	Bruni	PAOLO	1234	7-444-5555	
4	LAWSON	DOUGIE	3333	1-333-4444	
5	WICKS	GARY	4712	8-777-4444	
6	LAST1	FIRST1	D01/R01	8-111-1111	
7	MICKEY	GOOFY	1111	1234567890	
8	MIOTKE	PETER	4444	9-123-1234	
9	NICHOLS	GEOFF	2345	1-222-3333	
10	LAST2	FIRST2	D02/R02	8-111-2222	
11	LAST3	FIRST3	D03/R03	8-111-3333	
12	LAST5	FIRST5	D05/R05	8-111-5555	
13	ALAMEDA	ALMADEN	1234567	8-911-9115	
14	LAST4	FIRST4	D04/R04	8-111-4444	

Figure 9-18 Query result

For more information about using IMS Open Database with the Installation Verification Program, go to the following website:

<http://www-03.ibm.com/support/techdocs/atsmastr.nsf/WebIndex/WP101809>

To the default LASTx FIRSTx in this example, add a few columns by either using the SQL Scripts or by modifying the table as described in Figure 9-8 on page 355.

Many new administrators are familiar with running SQL statements but need to know how a SQL statement is translated into DL/I calls. Therefore, a SQL to DL/I Call Translator is available in IMS. To access it, go to IMS Explorer and open the SQL to DL/I Call Translation window, which is shown in Figure 9-19. In this example, we did a SELECT from the Phonebook Query and clicked **Translate**. The result is a GHU Get hold Unique Call loop that shows the same results.

SQL to DL/I Call Translation

Connection profile: I13B

Name of PSB to query: DFSIVP37

SQL statements: SELECT LASTNAME, FIRSTNAME, ZIPCODE, EXTENSION
FROM PHONEAP.PHONEBOOK

DL/I calls: GHU A1111111
(LOOP)
GHN A1111111

Translate

Figure 9-19 SQL to DL/I translator

Another example of the translator is a delete row that looks like the GHU Delete Call, as shown in Figure 9-20.

The screenshot shows a window titled "SQL to DL/I Call Translation". It contains the following fields and text:

- Connection profile:** I13B
- Name of PSB to query:** DFSIVP37
- SQL statements:**

```
delete from "PHONEAP"."PHONEBOOK" where
"LASTNAME"='ALAMEDA '
```
- DL/I calls:**

```
GHU  A1111111 (A1111111EQALAMEDA  )
DLET

(LOOP)
GHN  A1111111 (A1111111EQALAMEDA  )
DLET
```
- Translate** button

Figure 9-20 Delete row

For more information about IBM IMS Enterprise Suite Explorer for Developers, you can view a web page about the lab about IMS Explorer that was created by Suzie Wendler and Ken Blackman and presented at the SHARE 2013 conference in San Francisco:

<https://share.confex.com/share/120/webprogram/Session12564.html>



IMS Data Provider for Microsoft .NET

Microsoft .NET is a popular development platform. Over the last few years, the market has created a demand for interoperability solutions between .NET applications and IBM IMS. As described in Chapter 5, “SOAP application technology” on page 169, IMS SOAP Gateway provides a way to access IMS applications from .NET. However, until recently, there was no easy way to access the IMS data itself from .NET.

With the introduction of *IBM IMS Data Provider for Microsoft .NET (IMS Data Provider)*, a future component of the IBM IMS Enterprise Suite for z/OS V3.1 (5655-TDA), this access will become possible. This solution allows streamlined and standards-based way of accessing IMS databases from any .NET applications using SQL.

This chapter covers the following topics:

- ▶ Overview of the IBM IMS Data Provider for Microsoft .NET
- ▶ Programming applications with IBM IMS Data Provider for Microsoft .NET

Statement of direction:

IBM intends, at a future time, possibly through its support and service processes, to make available IMS Data Provider for Microsoft .NET as part of the IMS Enterprise Suite for Distributed Systems offering.

IBM's statements regarding its plans, directions, and intent are subject to change or withdrawal without notice at IBM's sole discretion. Information regarding potential future products is intended to outline our general product direction and it should not be relied on in making a purchasing decision. The information mentioned regarding potential future products is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. Information about potential future products may not be incorporated into any contract. The development, release, and timing of any future features or functionality described for our products remains at our sole discretion.

10.1 Overview of the IBM IMS Data Provider for Microsoft .NET

This section provides an overview of the IMS Data Provider for Microsoft .NET.

This section includes the following items:

- ▶ ADO.NET application development
- ▶ IMS Data Provider prerequisites and system requirements
- ▶ IMS Data Provider architecture

10.1.1 ADO.NET application development

In recent years, Microsoft has been promoting a new software development platform for Windows, known as the *.NET Framework*. The .NET Framework is Microsoft's replacement for Component Object Model (COM) technology. The following points highlight the key .NET Framework features:

- ▶ You can code .NET applications in over forty different programming languages. The most popular languages for .NET development are C# and Visual Basic .NET.
- ▶ The .NET Framework class library provides the building blocks with which you build .NET applications. This class library is language neutral and provides interfaces to operating system and application services.
- ▶ Your .NET application (regardless of language) compiles into Intermediate Language (IL), a type of bytecode.
- ▶ The Common Language Runtime (CLR) is the heart of the .NET Framework, compiling the IL code dynamically and then running it. In running the compiled IL code, the CLR activates objects, verifies their security clearance, allocates their memory, runs them, and cleans up their memory after the run is finished.

Through these features, the .NET Framework facilitates a wide variety of application implementations (for example, Windows forms, web forms, and web services), rapid application development, and secure application deployment. COM and COM+ proved to be inadequate or cumbersome for all these features.

The .NET Framework provides extensive data access support through ADO.NET. ADO.NET supports both *connected* and *disconnected* access. The key component of disconnected data access in ADO.NET is the *DataSet* class, instances of which act as a database cache that is in your application's memory.

For both connected and disconnected access, your applications use databases through a *data provider*. Various database products include their own .NET data providers, including IMS.

A .NET data provider features implementations of the following basic classes:

Connection	Establishes and manages a database connection.
Command	Runs an SQL statement against a database.
DataReader	Reads and returns result set data from a database.
DataAdapter	Links a DataSet instance to a database. Through a DataAdapter instance, the DataSet can read and write database table data.

Microsoft provides three data providers: The SQL .NET Data Provider, the OLE DB .NET Data Provider, and the ODBC .NET Data Provider. The IBM IMS Data Provider for Microsoft .NET is a high performance, managed ADO.NET data provider that is created specifically for IMS database systems.

10.1.2 IMS Data Provider prerequisites and system requirements

Before deployment, you must build your .NET application, which you can do with either *Visual Studio* or the *command line*. Computers that you use to build .NET applications and computers where you deploy .NET applications must have a supported version of the Windows operating system, in addition to other software.

- ▶ Build systems
 - Windows operating system (Windows XP or Windows 7)
 - Visual Studio (2010 or later)
 - .NET Framework Redistributable Package
 - .NET Framework Software Development Kit
- ▶ Deployment systems
 - Windows operating system
 - .NET Framework Redistributable Package

With the IMS .NET Provider, your .NET applications can access IMS DB system, Version 13 or later. IMS Catalog, ODBM, and Native SQL features must be configured.

Before you install the IBM IMS Data Provider for Microsoft .NET, you must already have of .NET Framework Version 4.0 or later.

10.1.3 IMS Data Provider architecture

Until recently, .NET-based IMS customers who needed access to IMS data from .NET applications were forced to do the following tasks:

- ▶ Set up a replication environment where IMS data is copied to a .NET accessible database server.
- ▶ Implement data proxy solutions that do not offer direct connectivity.
- ▶ Increase the code path by implementing bridge solutions.

All of these alternative paths mean either higher development costs, higher management costs, or higher runtime costs.

IMS Data Provider for Microsoft .NET, the newest member of the IMS Enterprise Suite, provides standard SQL access to IMS data from Microsoft .NET applications using a standards-based approach that enables existing tool and application support. IMS Data Provider for Microsoft .NET enables transparent and direct support of IMS data access from .NET. It simplifies development of Microsoft .NET applications (written in C#, Visual Basic, and other ADO.NET compliant languages) accessing IMS and allows reading and manipulating IMS data from Microsoft .NET applications.

This access eliminates the need for intermediate steps/tools (DB2 stored procedures, web services, IBM InfoSphere® Classic Federated Server, and other third-party products) for access to IMS databases.

.NET applications access IMS database through the *IMS Data Provider* (outlined in red dots in Figure 10-1).

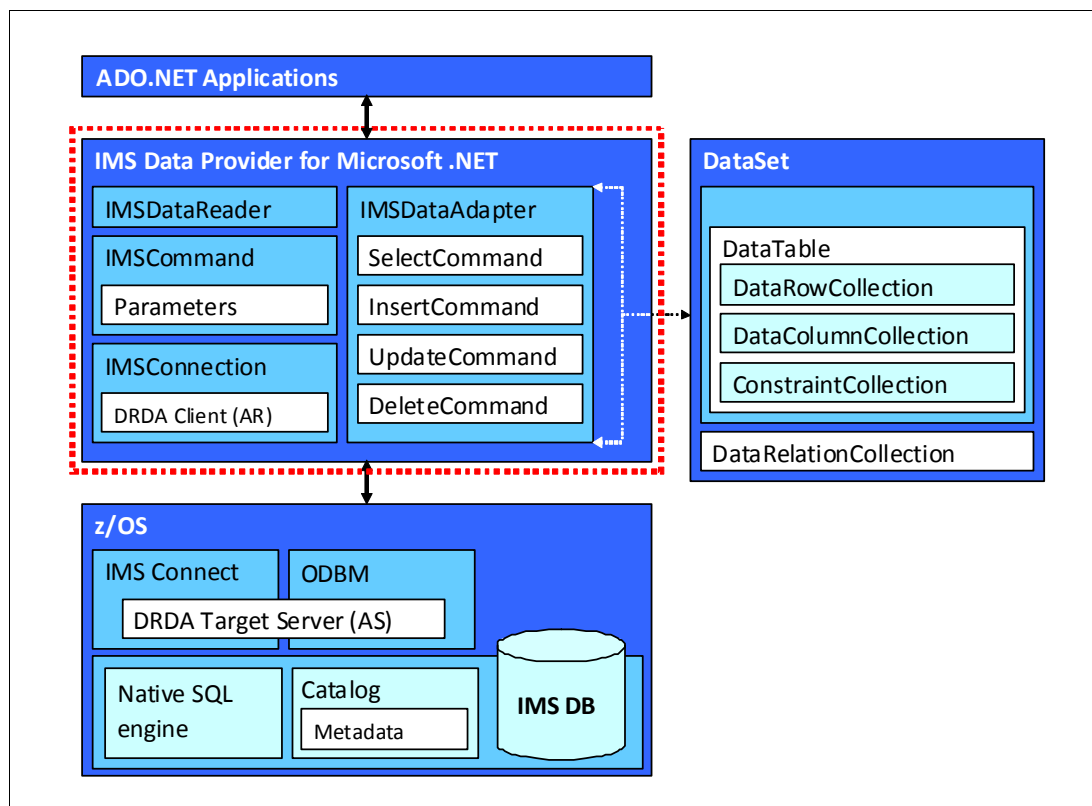


Figure 10-1 Overview of IMS .NET Data Provider architecture

The IMS Data Provider has been designed for data manipulation and fast access to data:

- ▶ The *Connection* object provides connectivity to a data source.
- ▶ The *Command* object enables access to database commands to return data, modify data, and send or retrieve parameter information.
- ▶ The *DataReader* provides a high-performance stream of data from the data source.
- ▶ The *DataAdapter* provides the bridge between the *DataSet* object and the data source. The *DataAdapter* uses *Command* objects to run SQL commands at the data source to both load the *DataSet* with data and reconcile changes that were made to the data in the *DataSet* back to the data source.

The API for the IMS Data Provider is immediately familiar to .NET database developers because a *data provider* is a standard type of component.

Example 10-1 shows an example of a simple C# application connecting to IMS and retrieving data.

Example 10-1 Simple C# application connecting to IMS and retrieving data

```
using IBM.Data.IMS;

static void IMSReader()
{
    // New Connection object, uses connection string to configure connection
    properties
```

```

    IMSConnection conn = new IMSConnection("Data
source=myims,5555;Database=BMP255");
    // Establish connection to IMS database
    conn.Open();

    // Specify SQL query in the Command object
    IMSCommand cmd1 = new IMSCommand("SELECT * FROM HOSPITAL", conn);

    // Execute query via the DataReader object
    IMSDataReader reader = cmd1.ExecuteReader();

    // Iterate through results and output on the screen
    while (reader.Read())
    {
        Console.Write(reader.GetString(0) + "\t");
        Console.WriteLine();
    }

    // Close reader
    reader.Close();
    // Close connection
    conn.Close();
}

```

You might notice that this code looks similar to a Java JDBC equivalent.

10.2 Programming applications with IBM IMS Data Provider for Microsoft .NET

This section provides some examples about how IMS Data Provider can be used to connect to IMS, access data by running SQL statements, process metadata, and activate a trace.

This section covers the following topics:

- ▶ Generic coding with the ADO.NET common base classes
- ▶ Connecting to IMS from an application
- ▶ Connection pooling with the IMS Data Provider
- ▶ SQL data type representation in ADO.NET database applications
- ▶ Running SQL statements
- ▶ Reading result sets
- ▶ Connected versus Disconnected Modes
- ▶ Choosing between DataReader and DataSet
- ▶ Using IMSDataAdapter and DataSet for Disconnected data processing
- ▶ Processing metadata with IMS Data Provider
- ▶ Tracing IMS Data Provider

10.2.1 Generic coding with the ADO.NET common base classes

Proprietary IMS prefixed class names can be a problem. What if you want to write generic code, not tied to a particular database, or maybe supporting several different database flavors at once? For this purpose, .NET Framework outlines the *generic coding* or *factory-based* interface that is supported by IMS Data Provider.

The .NET Framework features a namespace that is called *System.Data.Common*, which features a set of base classes that can be shared by any .NET data provider. This facilitates a generic ADO.NET database application development approach, featuring a constant programming interface across different databases. When you use this technique, proprietary class names, such as *IMSCConnection*, are replaced with common names, such as *DbConnection*.

Example 10-2 shows a C# example that demonstrates the generic approach.

Example 10-2 Generic ADO.NET approach

```
static void GenericReader()
{
    DbProviderFactory factory = DbProviderFactories.GetFactory("IBM.Data.IMS");
    DbConnection conn = factory.CreateConnection();
    DbConnectionStringBuilder sb = factory.CreateConnectionStringBuilder();

    conn.ConnectionString = sb.ConnectionString;
    conn.Open();

    DbCommand cmd = new DbCommand("SELECT * FROM HOSPITAL", conn);
    DbDataReader reader = cmd.ExecuteReader();
    while (reader.Read())
    {
        Console.Write(reader.GetInt32(0) + "\t");
        Console.WriteLine(reader.GetString(1) + "\t");
    }
    reader.Close();
    conn.Close();
}
```

10.2.2 Connecting to IMS from an application

When you use IBM Data Server Provider for .NET, a database connection is established through the *IMSCConnection* class.

To connect to IMS, complete the following steps:

1. Create a string that stores the connection parameters. Here is the format for a typical connection string format:

```
Server=<ip address/hostname/localhost>:<port number>;
Database=<PSB name>;
Datastore=<Datastore value>;
UID=<userID>;
PWD=<password>;
Connect Timeout=<Timeout value>;
Pooling=<true or false>;
```

2. Pass the connection string to the *IMSCConnection* constructor. For example:

```
String connectString = "Database=SAMPLE";
IMSCConnection conn = new IMSCConnection(connectString);
```

3. Use the *IMSCConnection* object's **Open** method to formally connect to the database identified in the connection string:

```
conn.Open();
```

10.2.3 Connection pooling with the IMS Data Provider

When a connection is first opened against an IMS database, a connection pool is created. As connections are closed, they enter the pool, ready to be reused within the same process by other applications that need connections.

The IMS Data Provider uses a normalized set of connection string attributes for determining the connection pool. By using normalized attributes, the chances of an application reusing connections are increased.

The IMS Data Provider enables connection pooling by default.

Note: You can turn off connection pooling by using the `Pooling=false` connection string keyword/value pair.

You can control the behavior of the connection pool by setting connection string keywords for the following items:

- ▶ The minimum and maximum pool size (`MinPoolSize` and `MaxPoolSize`)
- ▶ The length of time that a connection can be idle before it is returned to the pool (`ConnectionLifetime`)

10.2.4 SQL data type representation in ADO.NET database applications

ADO.NET database applications can reference IMS SQL data type values as parameter values to be used as part of SQL statement execution and as variables, but the appropriate IMS Data Provider data type values and .NET Framework data type values must be used to ensure that there is no truncation or loss of data when accessing or retrieving the values.

For specifying parameter values to be used as part of a SQL statement to be run, IMS Data Provider objects must be used. The `IMSPParameter` object is used to represent a parameter to be added to an `IMSCCommand` object, which represents a SQL statement. When specifying the data type value for the parameter, the IMS Data Provider data type values available in the `IBM.Data.IMS.IMSType` namespace must be used.

Example 10-3 shows an example of using `IMSPParameter` and `Prepare` methods in C#.

Example 10-3 IMSPParameter and Prepare methods in C#

```
static void IMSCCommandPrepareEx()
{
    IMSConnection conn = new IMSConnection("Data
source=myims,5555;Database=BMP255");
    conn.Open();

    IMSCCommand command = new IMSCCommand(null, conn);

    // Create and prepare an SQL statement.
    command.CommandText = "INSERT INTO PATIENT (PATNAME, SEX) VALUES (?, ?)";
    IMSPParameter patParam = new IMSPParameter("PATNAME", IMSType.VARCHAR, 100);
    IMSPParameter sexParam = new IMSPParameter("SEX", IMSType.VARCHAR, 100);
    patParam.Value = "Evgeni Liakhovich";
    sexParam.Value = "M";
    command.Parameters.Add(patParam);
    command.Parameters.Add(sexParam);
}
```

```

//Creates a prepared (or compiled) version of the command on the data source
command.Prepare();
command.ExecuteNonQuery();

conn.Close();
}

```

Table 10-1 shows the mappings across IMS data types, DbType data types, and Microsoft .NET Framework types.

Table 10-1 Mappings across IMS, DbType, and Microsoft .NET Framework types

IMS data type	DbType data type	.NET data type
TINYINT	Byte	Byte
INTEGER	Int32	Int32
CHAR	StringFixedLength	String
DOUBLE	Double	Double
FLOAT	Single	Single
BIT	Boolean	Boolean
BIGINT	Int64	Int64
SMALLINT	Int16	Int16
VARCHAR	String	String
PACKEDDECIMAL	Decimal	Decimal
ZONEDDECIMAL	Decimal	Decimal
DATE	Date	DateTime
TIME	Time	DateTime
BINARY	Binary	Byte
BYTE	SByte	Byte
ULONG	UInt64	UInt64
UINT	UInt32	UInt32
USHORT	UInt16	UInt16
UBYTE	Byte	Byte
OTHER	Object	Object

10.2.5 Running SQL statements

IMS Data Provider supports the same subset of standard SQL 2008 as the IMS Universal Driver. In fact, the actual SQL processing engine for all of IMS solutions converged into a single location: The Native SQL feature of IMS V13. Close proximity of this new SQL engine and IMS database ensures faster query processing, including aggregate queries.

When you use the IMS Data Provider, the running of SQL statements is done through a `IMSCCommand` class using its methods `ExecuteReader()` and `ExecuteNonQuery()`, and its properties `CommandText` and `Transaction`.

For SQL statements that produce output, the `ExecuteReader()` method should be used and its results can be retrieved from an `IMSDataReader` object. For all other SQL statements, the method `ExecuteNonQuery()` should be used. The `Transaction` property of the `IMSCCommand` object should be initialized to an `IMSTransaction` object. An `IMSTransaction` object is responsible for rolling back and committing database transactions.

Note: For optimal performance, qualify segment names with PCB names, if more than one PCB exists in a PSB (for example, both `SELECT * FROM PATIENT` and `SELECT * FROM PCB01.PATIENT` work, but the second query usually yields better performance).

Example 10-4 shows a sample of running a `SELECT` statement in C#.

Example 10-4 SELECT statement in C#

```
IMSCCommand cmd = conn.CreateCommand();
IMSTransaction trans = conn.BeginTransaction();
cmd.Transaction = trans;
cmd.CommandText = "SELECT deptnumb, location " +
                  " FROM pcb01.org WHERE deptnumb < 25";

IMSDataReader reader = cmd.ExecuteReader();
```

Example 10-5 shows an example of running an `UPDATE` statement in C# (use the same code for `INSERT` and `DELETE` statements).

Example 10-5 UPDATE statement in C#

```
IMSCCommand cmd = conn.CreateCommand();
IMSTransaction trans = conn.BeginTransaction();
cmd.Transaction = trans;
cmd.CommandText = "UPDATE pcb01.patient SET PATNUM = 25 " +
                  " WHERE PATNAME='Mauricio Adames'";

cmd.ExecuteNonQuery();
```

The application creates an `IMSTransaction` object by calling `BeginTransaction` on the `IMSConnection` object. All subsequent operations that are associated with the transaction (for example, committing or aborting the transaction) are performed on the `IMSTransaction` object.

After your application performs a database transaction, you must either roll it back or commit it. This is done through the `Commit()` and `Rollback()` methods of an `IMSTransaction` object.

Example 10-6 shows an example of rolling back or committing a transaction in C#.

Example 10-6 Rolling back or committing a transaction in C#

```
trans.Rollback();
...
trans.Commit();
```

10.2.6 Reading result sets

When using the IMS Data Provider, the reading the result sets is done through a *IMSDataReader* object. The *IMSDataReader* method, **Read()**, is used to advance to the next row in the result set.

The methods **GetString()**, **GetInt32()**, **GetDecimal()**, and other methods for all of the available data types are used to extract data from the individual columns of output. The *IMSDataReader* **Close()** method is used to close the *IMSDataReader* object, which should always be done when reading the output is finished.

Example 10-7 shows an example of reading a result set in C#.

Example 10-7 Reading a result set in C#

```
Int16 deptnum = 0;
String location="";

// Output the results of the query
while(reader.Read())
{
    deptnum = reader.GetInt16(0);
    location = reader.GetString(1);
    Console.WriteLine("    " + deptnum + " " + location);
}
reader.Close();
```

10.2.7 Connected versus Disconnected Modes

IMS Data provider supports both *Connected* and *Disconnected* data access modes. The Connected mode uses the *DataReader* class and allows direct data manipulation and fast, forward-only, read-only access to data.

IMS Data Provider also serves as a bridge between a data source and an ADO.NET *DataSet* interface (shown in Figure 10-1 on page 370). The *DataSet* object is central to supporting disconnected, distributed data scenarios with ADO.NET. The *DataSet* object is a memory-resident representation of data that provides a consistent relational programming model regardless of the data source. The Disconnected architecture allows fetching data from IMS into a local *DataSet*, manipulating data without holding database locks, and committing all changes at once to IMS when all work is finished.

10.2.8 Choosing between DataReader and DataSet

When you decide whether your application should use a *DataReader* or a *DataSet*, consider the type of functionality that your application requires. Use a *DataSet* to do the following tasks:

- ▶ Cache data locally in your application so that you can manipulate it. If you need only to read the results of a query, the *DataReader* is the better choice.
- ▶ Interact with data dynamically, such as binding to a Windows Forms control or combining and relating data from multiple sources.
- ▶ Perform extensive processing on data without requiring an open connection to IMS, which frees the connection to be used by other clients.

If you do not require the functionality that is provided by the `DataSet`, you can improve the performance of your application by using the `DataReader` to return your data in a forward-only, read-only manner. Although the `DataAdapter` uses the `DataReader` to fill the contents of a `DataSet` (see “Updating data sources with `IMSDDataAdapter`” on page 377), by using the `DataReader`, you can boost performance because you save memory that would be consumed by the `DataSet`, and avoid the processing that is required to create and fill the contents of the `DataSet`.

10.2.9 Using `IMSDDataAdapter` and `DataSet` for Disconnected data processing

The *`IMSDDataAdapter`* serves as a bridge between a `DataSet` and database for retrieving and saving data. The `IMSDDataAdapter` provides this bridge by using the `Fill` method to load data from the database into the `DataSet`, and by using the `Update` method to send changes that are made in the `DataSet` back to the database.

The `IMSDDataAdapter` also includes the `SelectCommand`, `InsertCommand`, `DeleteCommand`, and `UpdateCommand` properties to facilitate the loading and updating of data.

Updating data sources with `IMSDDataAdapter`

The `Update` method of the `IMSDDataAdapter` is called to resolve changes from a `DataSet` back to the data source. The `Update` method, like the `Fill` method, takes as arguments an instance of a `DataSet`, and an optional `DataTable` object or `DataTable` name. The `DataSet` instance is the `DataSet` that contains the changes that were made, and the `DataTable` identifies the table from which to retrieve the changes. If no `DataTable` is specified, the first `DataTable` in the `DataSet` is used.

When you call the `Update` method, the `DataAdapter` analyzes the changes that were made and runs the appropriate command (`INSERT`, `UPDATE`, or `DELETE`). When the `DataAdapter` encounters a change to a `DataRow`, it uses the `InsertCommand`, `UpdateCommand`, or `DeleteCommand` to process the change. This allows you to maximize the performance of your application by specifying command syntax at design time. You must explicitly set the commands before calling `Update`. If `Update` is called and the appropriate command does not exist for a particular update (for example, no `DeleteCommand` for deleted rows), an exception is thrown.

Generating commands with `CommandBuilder`

If your `DataSet` was filled from a single database table, you can take advantage of the `IMSCCommandBuilder` object to automatically generate the `DeleteCommand`, `InsertCommand`, and `UpdateCommand` of the `IMSDDataAdapter`.

The *`IMSCCommandBuilder`* must run the `SelectCommand` to return the metadata that is necessary to construct the `INSERT`, `UPDATE`, and `DELETE` SQL commands. As a result, an extra trip to the data source is necessary, and this can hinder performance. To achieve optimal performance, specify your commands explicitly rather than using the `IMSCCommandBuilder`.

10.2.10 Processing metadata with IMS Data Provider

Database *metadata* (or *schema*) is used to describe the design and specification of the data structures in the database. IMS Data Provider offers two ways to read and process IMS metadata. First, there is the `GetSchemaTable` method of the `IMSDDataReader` class. This method provides only metadata for the result set that is associated with the `IMSDDataReader` object. Second, there are `GetSchema` methods of the `IMSConnection` object that provide collections of metadata about the entire database that is associated with the `IMSConnection` object.

Using the IMSDataReader.GetSchemaTable method for the result set metadata

The **GetSchemaTable** method returns a **DataTable** that describes the column metadata of the **IMSDataReader** object. A row is returned for every column in the result set. The field names that are returned by **GetSchemaTable** are listed in Table 10-2.

Table 10-2 *GetSchemaTable* method results

Field name	Description
ColumnName	The name of the column.
ColumnOrdinal	The ordinal of the column.
ColumnSize	The maximum length of a value in the column.
NumericPrecision	If the IMS Data Type is a numeric data type, the maximum precision of the column; otherwise, null.
NumericScale	If the IMS Data Type is decimal, the number of digits to the right of the decimal point; otherwise, null.
DataType	A data type that maps to the IMS Data Type.
ProviderType	The IMSType enumeration.
IsReadOnly	True if the column cannot be modified; otherwise, false.
BaseSchemaName	The name of the schema (PCB) in the database that contains the column.
BaseCatalogName	The name of the catalog (PSB) in the database that contains the column.
BaseTableName	The name of the table or view in the database that contains the column.
BaseColumnName	The name of the column in the database, if possible. This name might be different from the column name that is returned in the ColumnName column if you used an alias.

Using the IMSConnection.GetSchema method for the database metadata

Obtaining schema information from a database is accomplished by using *schema discovery*. Schema discovery allows applications to request that a data provider finds and returns information about the schema of a given database. Different database schema elements, such as tables, columns, and stored procedures, are exposed through *schema collections*. Each schema collection contains various schema information.

The **GetSchema** method and its overloads return a **DataTable** with metadata for the data source that is associated with this **IMSConnection** instance.

The schema collections that are supported by the IMS Data Provider and returned by **GetSchemaTable** are listed in Table 10-3.

Table 10-3 *GetSchemaTable* method results

Collection name	Description
MetaDataCollections	A list of the metadata collections that are supported by the IMS Data Provider
Restrictions	For each metadata collection, a list of qualifiers that can be used to restrict the scope of the requested metadata
Tables	A list of the tables in the data source that are associated with this IMSConnection instance
Columns	A list of the table columns in the data source that are associated with this IMSConnection instance
Schemas	A list of the schemas in the data source that are associated with this IMSConnection instance

Example 10-8 shows an example of using the **GetSchema** method to display database schema information.

Example 10-8 *Using the GetSchema method to display database schema information*

```
static void GetSchemaDemo()
{
    IMSConnection conn = new IMSConnection("Data
source=myims,5555;Database=BMP255;")
    conn.Open();

    // Using GetSchema without parameters will return supported collections
    DataTable table = conn.GetSchema();
    DisplayData(table);

    // Get a table of restrictions
    table = conn.GetSchema(IMSMetaDataCollectionNames.Restrictions);
    DisplayData(table);

    // Get all schemas (PCBs) of a given PSB
    table = conn.GetSchema("Schemas");
    DisplayData(table);

    // Get information about the HOSPITAL segment
    string[] restrictions = new string[4];
    restrictions[2] = "HOSPITAL";
    table = conn.GetSchema(IMSMetaDataCollectionNames.Tables, restrictions);
    DisplayData(table);

    // Get information about specific column
    restrictions = new string[4];
    restrictions[0] = "BMP255";
    restrictions[1] = "PHDAMVAR";
    restrictions[2] = "WARD";
    restrictions[3] = "WARDNAME";
    table = conn.GetSchema(IMSMetaDataCollectionNames.Columns, restrictions);
```

```

        DisplayData(table);
    }

    private static void DisplayData(System.Data.DataTable table)
    {
        foreach (System.Data.DataRow row in table.Rows)
        {
            foreach (System.Data.DataColumn col in table.Columns)
            {
                Console.WriteLine("{0} = {1}", col.ColumnName, row[col]);
            }
            Console.WriteLine("=====");
        }
    }
}

```

10.2.11 Tracing IMS Data Provider

Function entry and exit trace points in the .NET public API are traced by using *System.Diagnostics.Trace*. Only first level calls are traced, for example, only the provider calls that are explicitly made by the customer's application are traced. Any calls to the public API made internally by the provider itself are not traced.

For the IBM.Data.IMS namespace, the TraceSwitch is named **IMSDataProviderTrace**:

```

<configuration>
  <system.diagnostics>
    <switches>
      <add name="IMSDataProviderTrace" value="1" />
    </switches>
  </system.diagnostics>
</configuration>

```

The TraceSwitch class uses the *System.Diagnostics.TraceLevel* enumeration to control the level of information that is traced; 0 - 4 represents Off, Error, Warning, Info, and Verbose, respectively.

Example 10-9 is an example of what the trace output for a simple program might look like.

Example 10-9 Trace output

```

Tracing for IMS Data Provider for Microsoft .NET has been enabled. TraceSwitch
Name : [IMSDataProviderTrace] Value : [Verbose] Assembly File Version :
[IBM.Data.IMS, Version=1.0.23.1, Culture=neutral,
PublicKeyToken=5512d16b327977a3].
Tue Aug 27, 2013 14:17:32.305 PM : [ENTRY] [ThreadId:10] [HashCode:5822459]
IMSConnection()
Tue Aug 27, 2013 14:17:32.311 PM :      Initializing Connection Pooling...
Tue Aug 27, 2013 14:17:32.313 PM : [ENTRY] IMSConnPool..ctor()
Tue Aug 27, 2013 14:17:32.314 PM : [ENTRY] [ThreadId:10] [HashCode:10]
IMSConnPool.GetIMSConnPool()
Tue Aug 27, 2013 14:17:32.317 PM :      _iRefCount = 1
Tue Aug 27, 2013 14:17:32.318 PM : [EXIT] IMSConnPool.GetIMSConnPool()
Tue Aug 27, 2013 14:17:32.318 PM : [EXIT] IMSConnPool..ctor()
Tue Aug 27, 2013 14:17:32.996 PM :      Building EXCSAT command...
Tue Aug 27, 2013 14:17:32.996 PM : [ENTRY] DRDARRequest.buildEXCSAT()
Tue Aug 27, 2013 14:17:33.001 PM : [EXIT] DRDARRequest.buildEXCSAT()

```

```
Tue Aug 27, 2013 14:17:33.006 PM :          Flushing buffer...
Tue Aug 27, 2013 14:17:33.007 PM : [ENTRY] [ThreadId:10] [HashCode:687191]
DRDAT4Agent.sendRequest()
Tue Aug 27, 2013 14:17:33.007 PM : [ThreadId:10] [HashCode:687191] DRDA Request
Buffer :
```

(HEX)		(ASCII)	(EBCDIC)
0 1 2 3 4 5 6 7 8 9 A B C D E F	0123456789ABCDEF	0123456789ABCDEF	0123456789ABCDEF
004CD041 00010046 10410006 115EF1F0	.L.A...F.A...^..	.<}.....;10	:16
001A116D C1C4D4C9 D5C9C260 D2E5D5D3	...m.....`....	..._ADMINIB-KVNL	:32
E4F7F6E0 85A58785 9589001B 115AD6C4Z..	U76\evgeni...]OD	:48
60C9C3D6 D540F140 D6C460D6 C4C2D440	`....@.@..`....@	-ICON 1 OD-ODBM	:64
F16BF26B F3000711 47C4C6E2	.k.k....G...	1,2,3....DFS	:80



IMS mobile enablement solutions

IBM MobileFirst offers a comprehensive set of mobile products and services to increase efficiencies and gain a competitive advantage. As a mobile enterprise, you can attract new customers, and transform your business and IT infrastructure.

IBM conducted a study of 361 IT decision makers worldwide to identify and gain insights into the attributes of IT leaders in mobility. The research shows that mobile technology leaders are twice more likely to experience revenue growth of at least 10%.

Through IBM MobileFirst, IBM is providing companies with the essential tools to take advantage of new business opportunities that are being enabled by mobile.

This chapter describes the IMS mobile enablement integration solutions. It covers the mobile enablement solutions of IMS applications and databases in the following sections:

- ▶ IMS mobile enablement
- ▶ IBM Worklight
- ▶ IBM WebSphere DataPower

11.1 IMS mobile enablement

A growing, flexible workforce is rapidly adopting the usage of mobile devices for both personal and business activities. This shift presents many challenges as users seek access to the traditional enterprise IT resources with security, flexibility, and ease-of-use.

In today's mobile world, enterprises are transforming the way that they interact with their customers, partners, and employees by implementing mobile strategies. IBM has a new set of initiatives to accomplish the following tasks:

- ▶ Build, connect, and run a growing portfolio of mobile apps for customers, partners, and employees
- ▶ Manage and secure mobile applications and data on various mobile devices and operating systems
- ▶ Extend and transform the business to yield new opportunities and business models while extending existing business capabilities to mobile employees, customers, and partners

IMS has been investing in the mobile space for over a decade with the introduction of IMS Connect and other integration solutions. The new portfolio of mobile solutions by IBM opened new opportunities for IMS.

Figure 11-1 summarizes the IMS mobile solutions with IBM Worklight and IBM WebSphere DataPower.

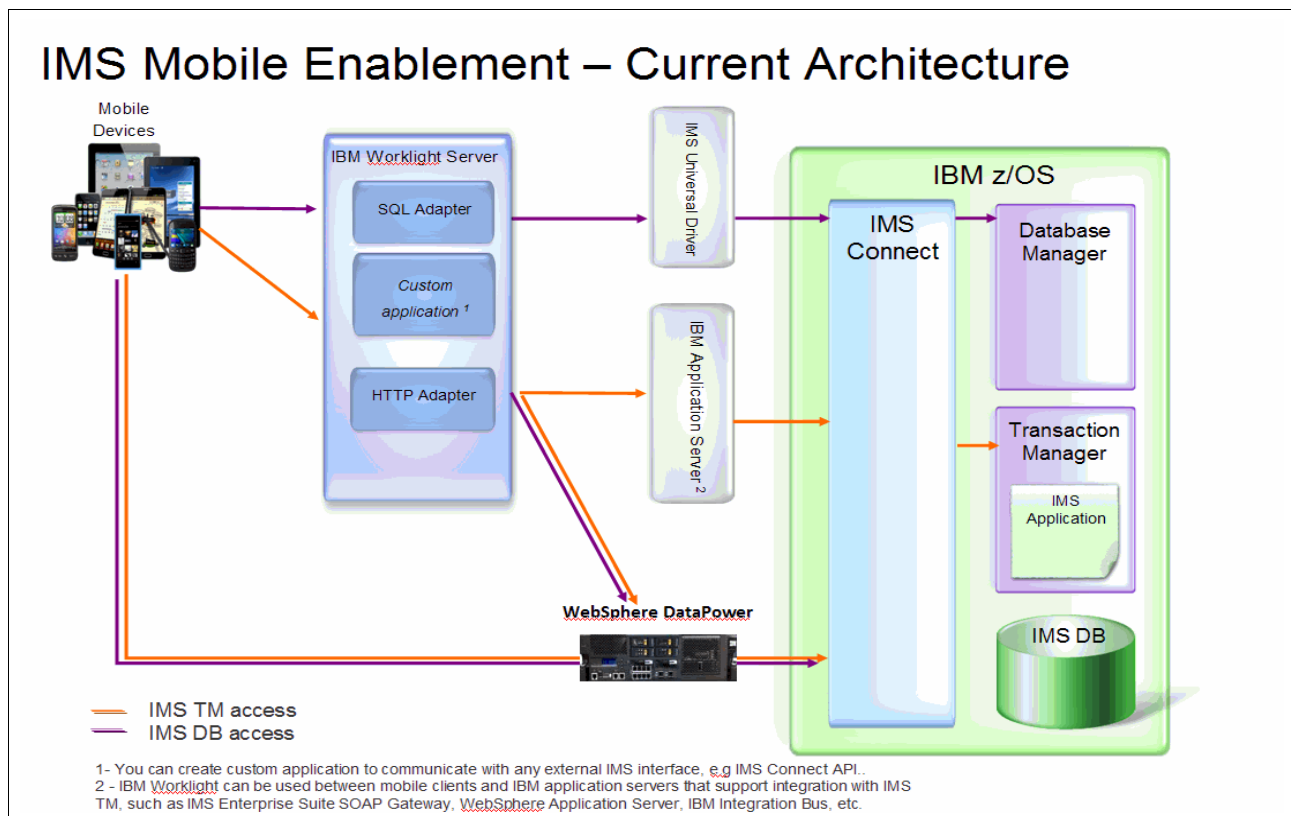


Figure 11-1 Overview of the connectivity options for IMS mobile enablement

In this setup:

- ▶ You can create a custom application to communicate with any external IMS interface, for example, the IMS Connect API.
- ▶ IBM Worklight can be used between mobile clients and IBM application servers that support integration with IMS, such as IBM IMS Enterprise Suite SOAP Gateway, IBM WebSphere Application Server, and IBM Integration Bus.

For more information about the IBM MobileFirst initiative, go to the following website:

<http://www.ibm.com/mobilefirst/us/en/why-ibm-for-mobile/>

11.2 IBM Worklight

IBM Worklight is a set of development and integration tools that provides an open, comprehensive, and advanced platform to build, run, and manage rich, cross-platform mobile applications, addressing many integration needs for mobile enablement.

IBM Worklight includes a suite of integration adapters that allow the Worklight platform to connect to back-end systems. Adapters that are provided with the product include the following ones:

- ▶ HTTP adapter (supports REST and SOAP)
- ▶ SQL adapter

For more information about Worklight, including tutorials and samples, go to the following website:

http://pic.dhe.ibm.com/infocenter/wrklght/v5r0m5/index.jsp?topic=%2Fcom.ibm.help.doc%2Fwl_home.html

For more information about Worklight adapters and procedures, see Modules 5, 5.1, 5.2, 5.3, and 6 at this website.

11.2.1 Accessing IMS transactions through Worklight HTTP Adapter

Worklight HTTP adapter works with RESTful and SOAP-based services. It can read structured HTTP sources (for example, IMS SOAP Gateway-based web services) and allows sending a GET or POST HTTP request and retrieves data from the response headers and body. The retrieved data can be in XML, HTML, JSON, or plain text formats.

IMS Enterprise Suite SOAP Gateway is a web services solution that enables IMS applications to interoperate outside of the IMS environment. It is compliant with the industry standards for web services, including SOAP/HTTP 1.1 and Web Services Description Language (WSDL) 1.1. By using the Worklight Server's HTTP/SOAP adapter, Mobile applications can interoperate with the IMS environment, as shown in Figure 11-2.

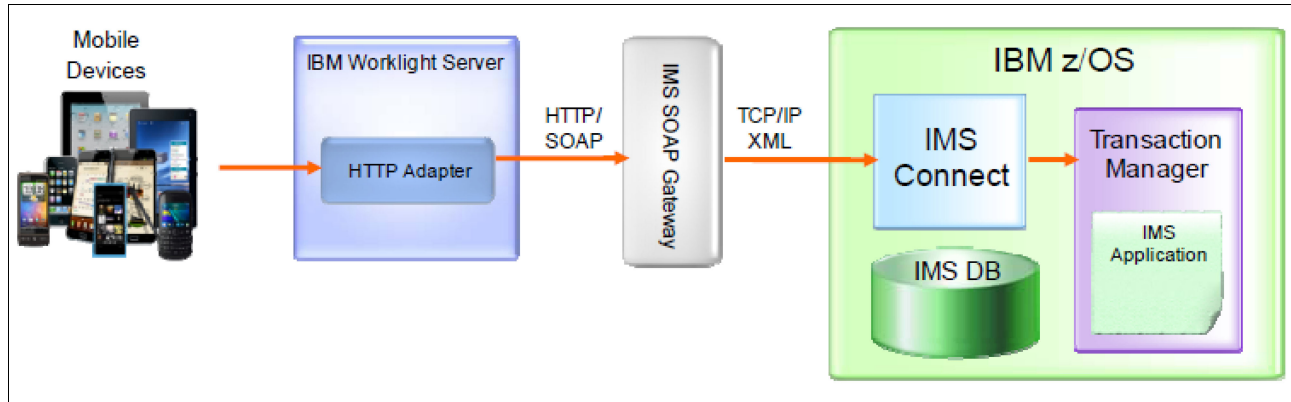


Figure 11-2 Mobile connectivity through Worklight Server and IMS SOAP Gateway

The HTTP adapter is configured through an XML file. To edit the adapter XML file, you need to complete the following steps:

1. Set the protocol to HTTP or HTTPS.
2. Set the HTTP domain of the domain part of the HTTP URL.
3. Set the TCP port.
4. Declare the required procedures.

Example 11-1 shows an HTTP adapter configuration file (IMSAdapter.XML).

Example 11-1 HTTP adapter configuration file (IMSAdapter.XML)

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wl:adapter xmlns:wl="http://www.worklight.com/integration"
xmlns:sql="http://www.worklight.com/integration/sql"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" name="IMSUniversalDriver">

  <displayName>IMSSOAPAdapter</displayName>
  <description>IMS SOAP Gateway Adapter</description>
  <connectivity>
    <connectionPolicy xsi:type="sql:HTTPConnectionPolicy">
      <protocol>http</protocol>
      <domain>zserveros.dfw.ibm.com</domain>
      <port>8085</port>
    </connectionPolicy>
    <loadConstraints maxConcurrentConnectionsPerNode="5"/>
  </connectivity>

  <procedure name="GetPhone"/>
</wl:adapter>
```

Procedures are implemented in the adapter JavaScript file. Procedures provide functions to the mobile applications. Functions typically include calling back-end systems (such as IMS) to retrieve data or to perform actions. In the procedure implementation, you must provide the web service URL and the input parameters in the form of a SOAP envelope.

Example 11-2 shows a JavaScript file that is configured to connect to the IMS SOAP Gateway based Phonebook web service.

Example 11-2 JavaScript file that is configured to connect to the IMS SOAP Gateway based Phonebook web service

```
function getPhone() {
    var request =
        '<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">'+
        '<s:Body xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">'+
        '<INPUTMSG xmlns="http://www.IMSPHBKI.com/schemas/IMSPHBKIInterface">'+
        '<in_ll>59</in_ll>'+
        '<in_zz>0</in_zz>'+
        '<in_trcd>IVTN0</in_trcd>'+
        '<in_cmd>DISPLAY</in_cmd>'+
        '<in_name1>LAST1</in_name1>'+
        '</INPUTMSG>'+
        '</s:Body>'+
        '</s:Envelope>';
    var input = {
method : 'post',
returnedContentType : 'xml',
path : '/imssoap/services/IMSPHBKService?wsdl',
body: {
    content: request.toString(),
    contentType: 'text/xml; charset=utf-8'
}
};
    return WL.Server.invokeHttp(input);
}
```

The response from running the script file in Example 11-2 is shown in Example 11-3.

Example 11-3 Response from the JavaScript file

```
{
  "Envelope": {
    "Body": {
      "OUTPUTMSG": {
        "out_cmd": "DISPLAY",
        "out_extn": "8-111-1111",
        "out_ll": "93",
        "out_msg": "ENTRY WAS DISPLAYED",
        "out_name1": "LAST1",
        "out_name2": "FIRST1",
        "out_segno": "0011",
        "out_zip": "D01\\R01",
        "out_zz": "768",
        "xmlns": "http:\\\\www.IMSPHBK0.com\\schemas\\IMSPHBK0Interface"
      }
    },
    "soapenv": "http:\\\\schemas.xmlsoap.org\\soap\\envelope\\"
  },
  "errors": [
  ],
}
```

```

    "info": [
    ],
    "isSuccessful": true,
    "statusCode": 200,
    "statusReason": "OK",
    "warnings": [
    ]
  }

```

11.2.2 Accessing IMS data through the Worklight SQL adapter

A Worklight SQL adapter can communicate with any SQL data source.

A JDBC connector driver for a specific database type must be downloaded separately by the developer and added to the `lib\` folder of a Worklight project. In the case of IMS, the IMS Universal JDBC Driver allows access to the IMS database through type-4 connectivity (IMS Version 12 and later and IMS Catalog are required for this function to work). For an overview, see Figure 11-3.

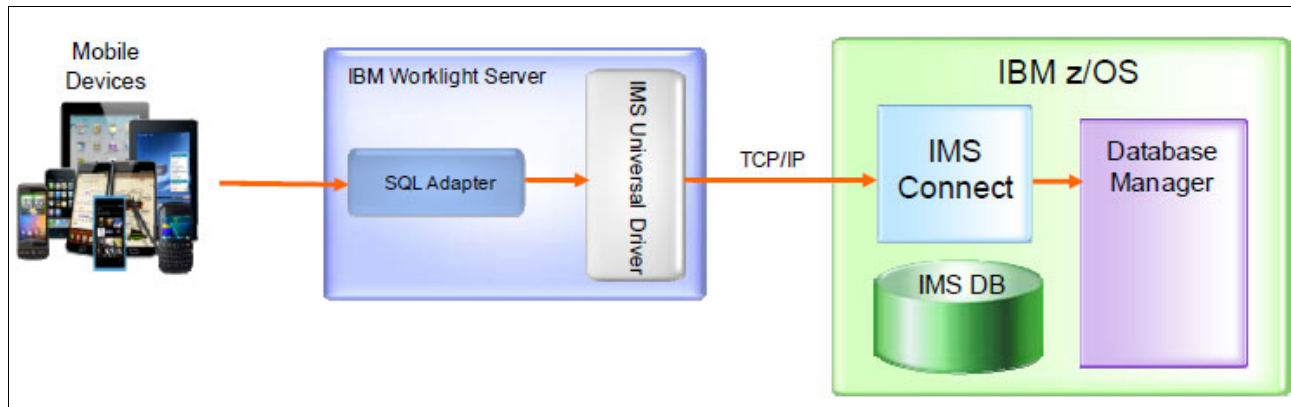


Figure 11-3 Accessing IMS from mobile devices through the Worklight SQL adapter

Similar to HTTP adapters, SQL adapter is configured through an XML file and procedures are implemented in the adapter JavaScript file. In the procedure implementation, you must provide an SQL query for the IMS database.

Example 11-4 shows a JavaScript file that is configured to query an IMS database.

Example 11-4 JavaScript file that is configured to query an IMS database

```

//Create SQL query
var getpartStatement = WL.Server.createStatement(
    "SELECT * FROM PCB01.PARTROOT WHERE PART_NO_EDIT = '?'";
);

//Invoke prepared SQL query and return invocation result
function GetPart(partnum){
    return WL.Server.invokeSQLStatement({
        preparedStatement : getpartStatement,
        parameters : ["partnum"]
    });
};

```

11.2.3 Using a custom Java application in Worklight to connect to IMS

Where JavaScript is not enough to implement certain functions, or a Java class exists, it is possible to use a custom Java application in Worklight. To use an existing Java library, add the JAR file to the server\lib folder of your Worklight Project. To add a custom Java application to your project, add a Java class file to the server/java folder in your Worklight project.

The Java application may communicate with the IMS Universal Driver, IBM application server, the IMS Connect API that is deployed as the JAR library, or directly with IMS.

Using a custom application opens new possibilities. Section 11.2.2, “Accessing IMS data through the Worklight SQL adapter” on page 388 described how to issue SQL queries to IMS through the Universal JDBC Driver. In your custom Java application, you can fully employ the JDBC API, for example, to examine database metadata.

Example 11-5 shows Java code that scans an IMS catalog.

Example 11-5 Java code that scans an IMS catalog

```
public static String ObtainTables(String psb)
{
    StringBuffer result = new StringBuffer();

    String url = "jdbc:ims://zserveros.dfw.ibm.com:7013/" + psb +
":dpsbOnCommit=true;fetchSize=0;";
    String user = "username";
    String password = "password";

    try {
        Connection conn = DriverManager.getConnection(url, user, password);
        DatabaseMetaData dbmd = conn.getMetaData();

        ResultSet sc = dbmd.getSchemas();
        while(sc.next()){
            for(int i=1; i<=sc.getMetaData().getColumnCount();i++)
                result.append(sc.getString(i)).append("|");
        }

        String[] types = {"TABLE"};
        ResultSet rs = dbmd.getTables(null,null,"%",types);
        while (rs.next())
            result.append(rs.getString("TABLE_NAME")).append("|");

        if(result.length() == 0)
            result.append("No records match the provided query");

        rs.close();
        conn.commit();

        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
        result.append(e.toString());
    }
}
```

```
        return result.toString();  
    }  
}
```

After your custom Java code is created and any required JAR files are added, you can reference it from your Worklight Adapter JavaScript as though it was written in Java, as shown in Example 11-6.

Example 11-6 Referencing the custom Java code

```
function GetTables(param) {  
    var arrayAsAString = com.worklight.customcode.UDCaller.ObtainTables(param);  
    realJsString = arrayAsAString + "  
    arrayFromJava = realJsString.split("|");  
    return {  
        result: arrayFromJava  
    };  
}
```

11.3 IBM WebSphere DataPower

IBM WebSphere DataPower is an appliance-based ubiquitous security and integration gateway. For more information about WebSphere DataPower, see Chapter 12, “IBM WebSphere DataPower and IMS integration” on page 395.

WebSphere DataPower provides the security, control, integration, and optimization that is needed for mobile workloads, such as the following ones:

- ▶ SSL offload
- ▶ Threat protection
- ▶ Rate limiting
- ▶ Validation and filtering
- ▶ Native XML and JSON support
- ▶ Authentication
- ▶ z/OS identity propagation
- ▶ Authorization
- ▶ OAuth 2.0
- ▶ Security token translation (for example, Security Assertion Markup Language (SAML))
- ▶ Content transformation
- ▶ Content-based routing
- ▶ Intelligent load distribution
- ▶ Response caching locally or to WebSphere DataPower XC10

WebSphere DataPower can play different roles in mobile enablement, providing a representational state transfer (REST) service façade, a DMZ proxy to secure a mobile network, and seamless enterprise integration for IBM Worklight, as shown in Figure 11-4.

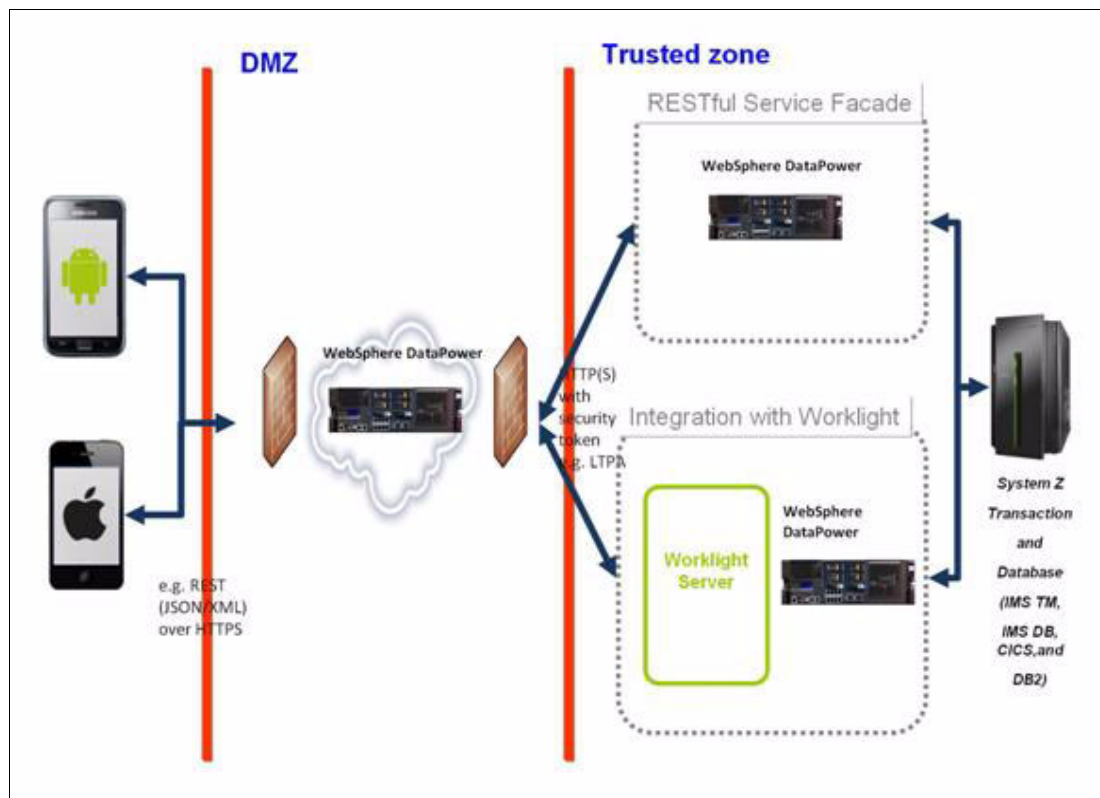


Figure 11-4 IMS mobile enablement with WebSphere DataPower

11.3.1 RESTful service facade

WebSphere DataPower is positioned to bridge Web 2.0 and SOA. It can service Web 2.0 requests, including REST (JSON) invocation, acting as a bridge to enterprise protocols, such as IMS Connect. As a service facade, WebSphere DataPower can expose enterprise systems RESTfully for mobile access with no changes to the back-end resources. For a simpler configuration, you can configure WebSphere DataPower to directly access enterprise assets.

For more information about configuring IMS and WebSphere DataPower environments to access an IMS transaction, see 12.3.4, “Inbound access to IMS transactions from an external client” on page 402; to access an IMS database, see 12.3.6, “Access to IMS databases” on page 421.

Figure 11-5 depicts a customer mobile demonstration that uses WebSphere DataPower XI52 to start an IMS application, which in turn performs an outbound callout to start a web service.

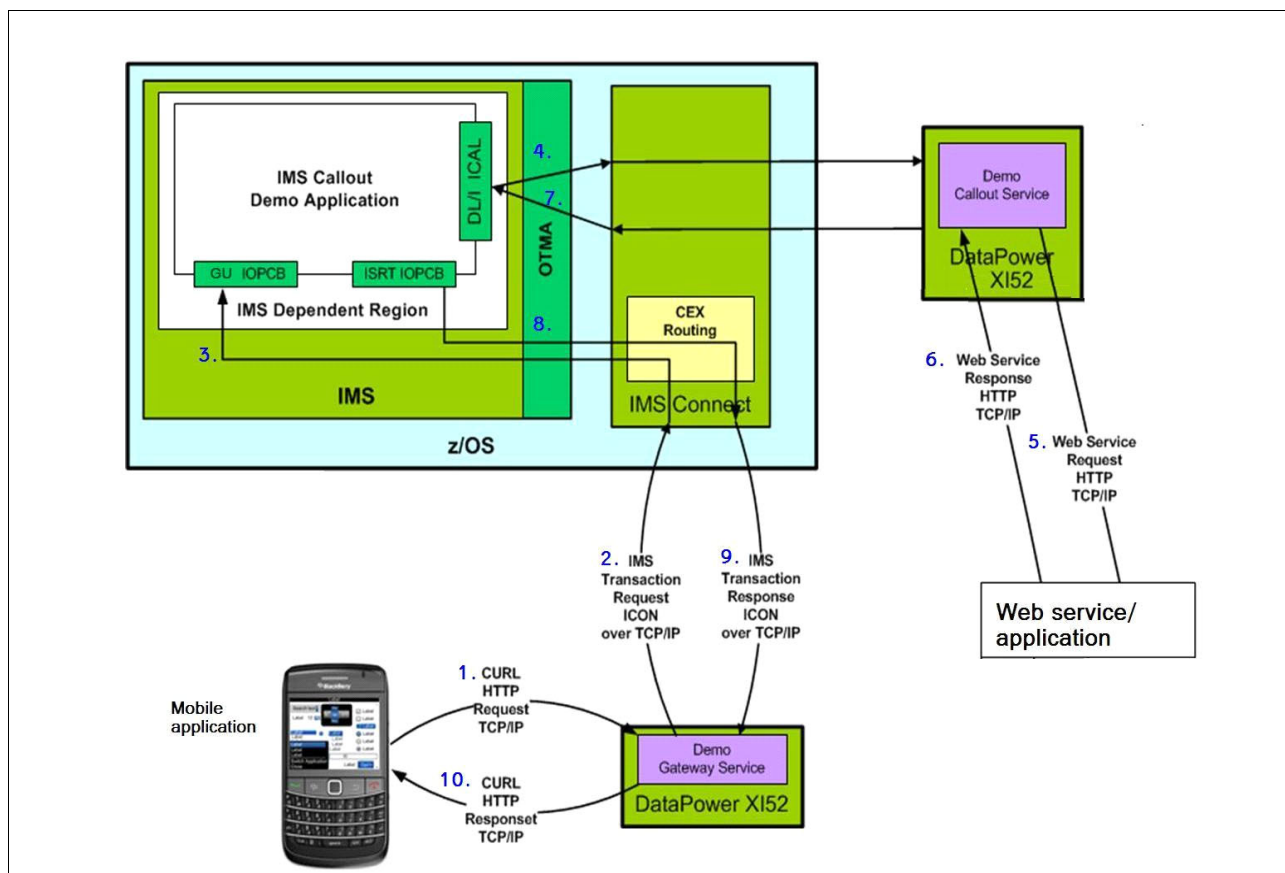


Figure 11-5 Mobile scenario with WebSphere DataPower starting an IMS application issuing an outbound callout for a web service

11.3.2 DMZ Proxy to secure a mobile network

WebSphere DataPower provides much-needed DMZ security for mobile traffic. When deployed in a DMZ, it can ensure connectivity to back-end components, such as the Worklight Server, use appropriate firewalls, proxies, or virtual private networks (VPNs), provide load balancing, and enforce security policies.

As shown in Figure 11-4 on page 391, WebSphere DataPower, in the DMZ, acts as the secure gateway to terminate inbound connections and provide user authentication/authorization offloading. After the request is trusted, WebSphere DataPower can translate it and communicate with enterprise systems directly or through the Worklight Server. Although the figure depicts two separate WebSphere DataPower Appliances in the trusted zone, both deployment scenarios can be achieved with a single appliance instance.

11.3.3 Seamless enterprise integration for IBM Worklight

WebSphere DataPower can also be the integration gateway for mobile traffic coming through IBM Worklight. It provides features such as message routing, caching, protocol mediation, security token/identity mapping, and any-any data transformation to preprocess the message into data formats that are native to and can be processed by an enterprise system.

For more information about configuring IMS and WebSphere DataPower environments to access an IMS transaction, see 12.3.4, “Inbound access to IMS transactions from an external client” on page 402; to access an IMS database, see 12.3.6, “Access to IMS databases” on page 421.

Figure 11-4 on page 391 depicts the end-to-end flows of a mobile device securely accessing enterprise systems through WebSphere DataPower, based on the various scenarios. With the light footprint that is provided by WebSphere DataPower Appliances and IBM Worklight, support for mobile devices can be introduced while still using many existing assets and infrastructures.

In summary, WebSphere DataPower allows IT organizations to keep pace with the technology advances that are presented by next generation mobile and web applications to meet their integration and security needs.



IBM WebSphere DataPower and IMS integration

The IBM WebSphere DataPower SOA appliances are a suite of purpose-built ubiquitous network devices that provide security, rapid data transformation, and integration and routing functions for large-scale service and mobile integration.

This suite of devices can perform the following task:

- ▶ Simplify business integration with your enterprise services and systems, including IMS
- ▶ Provide centralized governance and policy control
- ▶ Improve web and mobile facing application security

The appliances can be used in various user scenarios to enable security, control, integration, and optimized access for a range of workloads, including mobile, web, API, business-to-business, web services, SOA, and cloud.

This chapter provides a general overview of WebSphere DataPower, including business use cases and functional capabilities. It then provides the in-depth description of integration solutions with IMS, including provider scenarios, consumer scenarios (callout), and database access.

This chapter covers the following topics:

- ▶ WebSphere DataPower SOA Appliances introduction
- ▶ WebSphere DataPower capabilities
- ▶ WebSphere DataPower and IMS integration solutions

12.1 WebSphere DataPower SOA Appliances introduction

WebSphere DataPower SOA Appliances are security and integration XML-aware network gateways, which are built for simplified deployment and hardened security, bridging multiple protocols and performing conversions at wire speed. WebSphere DataPower is available both as a hardware appliance edition and virtual edition. WebSphere DataPower has over a decade of innovation, with 2000 worldwide installations and 10,000+ physical units sold.

WebSphere DataPower can be used across various scenarios:

- ▶ **DMZ Proxy:** WebSphere DataPower provides Demilitarized Zone (DMZ) security for web and mobile traffic. When it is deployed in a DMZ, it can ensure connectivity to back-end components, use the appropriate firewalls, proxies, or virtual private networks (VPNs), provide load balancing, and enforce security policies.
- ▶ **RESTful service façade:** WebSphere DataPower is positioned to bridge Web 2.0 and SOA. It can service Web 2.0 requests, including REST (JSON) invocation, acting as a bridge to enterprise assets on System z, such as IMS, CICS, DB2, and WebSphere Application Server for z/OS. As a service facade, WebSphere DataPower can expose enterprise systems RESTfully with no changes to the back-end resources.
- ▶ **Integration Service Gateway or Back-end Server for integration with IBM Worklight:** WebSphere DataPower can also be the integration gateway for mobile traffic coming through IBM Worklight. It provides features such as message routing, caching, protocol mediation, security token/identity mapping, and any-any data transformation to preprocess the message into data formats that are native to and can be processed by an enterprise system.

Figure 12-1 shows common use cases for WebSphere DataPower in the DMZ and the trusted domain.

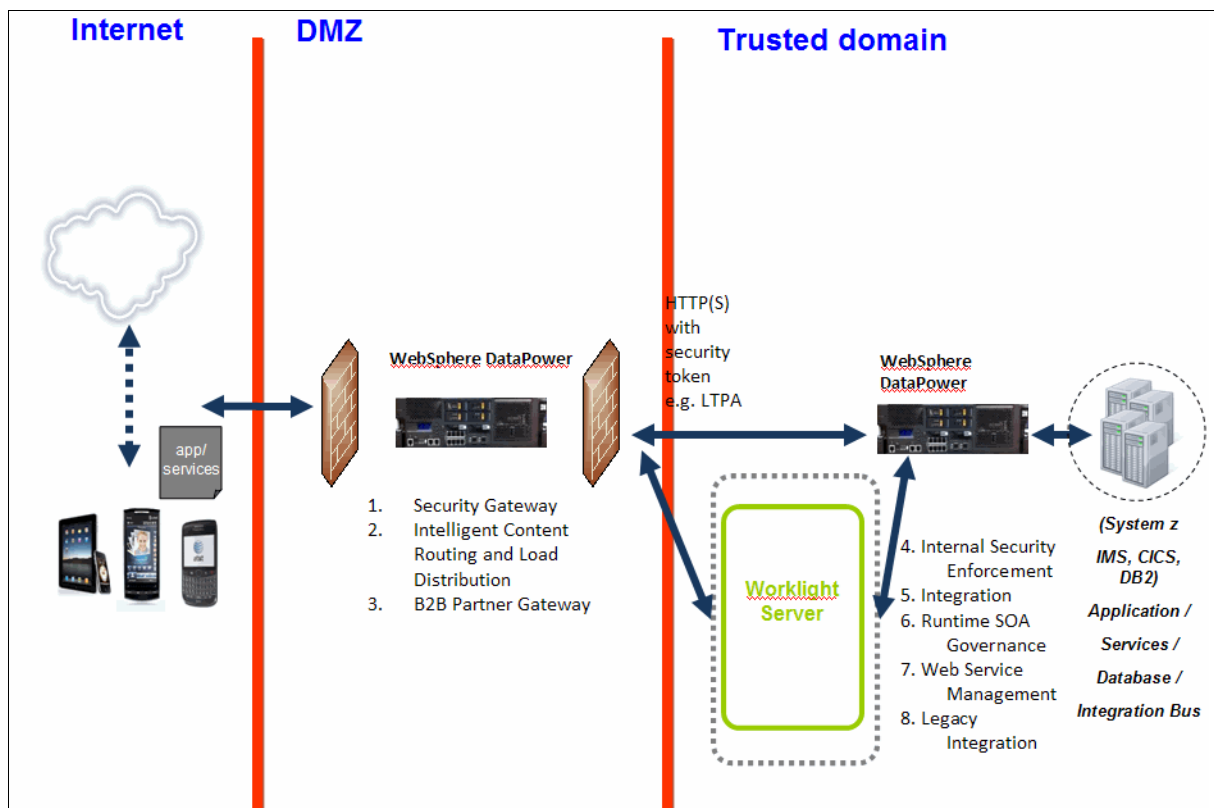


Figure 12-1 WebSphere DataPower common use cases

IBM WebSphere DataPower SOA Appliances V6.0 adds new processing capabilities for mobile, web, and application programming interface (API) traffic and introduces pattern-based configuration support that provides the following services:

- ▶ Helps secure, control, optimize, and integrate access to web, mobile, and API workloads.
- ▶ Helps secure mobile web traffic for IBM Worklight authentication integration for the IBM Worklight platform.
- ▶ A front-end proxy for IBM WebSphere Application Server with embedded router for WebSphere Application Server Network Deployment environments.
- ▶ Response caching and integration with WebSphere DataPower XC10
- ▶ Integration with IMS as a service provider, consumer, and also service dynamic SQL calls.
- ▶ Pattern-based configurations for creating and deploying common WebSphere DataPower configuration patterns.

12.2 WebSphere DataPower capabilities

This section describes some major capabilities of WebSphere DataPower. The product line provides different hardware models and firmware versions. Not all capabilities are available across all model/firmware versions.

WebSphere DataPower SOA Appliances are described at the following website:

<http://www.ibm.com/software/products/us/en/datapower/>

12.2.1 Any-to-any transformation engine

WebSphere DataPower supplies the runtime engine for any-to-any data transformation, which can be configured through the “WebSphere DataPower Multi-Protocol Gateway processing policy” transform action.

Tools: WebSphere Transformation Extender Design Studio

To address the need for tool-generated data transformation and data mapping, IBM offers WebSphere Transformation Extender Design Studio. WebSphere Transformation Extender can perform any-to-any data transforming and generates maps that can be deployed on any WebSphere DataPower appliance. Use WebSphere Transformation Extender maps to transform data from the format that is used in the IMS application to the format that is used by the distributed service/application.

For more information about WebSphere Transformation Extender, see the WebSphere Transformation Extender home page at the following website:

<http://www.ibm.com/software/products/us/en/wdatastagetx/>

12.2.2 Control

WebSphere DataPower facilitates the following control features:

- ▶ Service-level agreements
- ▶ Traffic control
- ▶ Message accounting
- ▶ Content-based routing
- ▶ Governance and management

12.2.3 Domains

The WebSphere DataPower domain allows entity separation, similar to the concept of LPARs on MVS systems or user environments in UNIX. There can be one default domain and multiple application domains. The default domain allows the administrator to control system-wide resources, such as network interfaces, users, and control objects. An application domain allows the users to have their own environment and maintain their own objects. A domain can be made visible to others to allow sharing of objects, for example, files.

12.2.4 Firewall

WebSphere DataPower supports different firewall features:

- ▶ The XML Firewall offers protection for externally exposed services against XML threats, heavy authentication, authorization, validation, DDOSs, virus, SQL injection attacks, XML node attacks, and more.
- ▶ The Web Application firewall offers protection for internet exposed web applications in the DMZ.

12.2.5 Firmware updates

WebSphere DataPower allows simple firmware upgrades through a GUI.

12.2.6 Interfaces

WebSphere DataPower provides two simple interfaces for configuration and management of the appliance:

- ▶ Command-line interface (CLI)
- ▶ Web GUI

The screen captures that are included in this chapter are taken from the web GUI interface.

12.2.7 Logging

WebSphere DataPower logging allows different level of logging configuration in each domain. It supports event subscription, event and object filters, log targets, and more.

12.2.8 Optimization

WebSphere DataPower facilitates the following optimization features:

- ▶ SSL and TLS offload
- ▶ Hardware accelerated crypto operations
- ▶ XSLT and XQuery acceleration
- ▶ JSON acceleration
- ▶ Connection pooling and offload
- ▶ Intelligent load distribution
- ▶ Caching: Locally and externally, for example, XC10

12.2.9 Monitoring

WebSphere DataPower allows SNMP monitoring using any industry standard operations management solution.

12.2.10 Multi-Protocol Gateway

A Multi-Protocol Gateway connects client requests that are transported over one or more protocols to a remote destination that uses the same or a different protocol. The Multi-Protocol Gateway supports the FTP, HTTP, HTTPS, IMS, WebSphere MQ, NFS, SFTP, WebSphere JMS protocols, and more.

A Multi-Protocol Gateway is required for communication with your IMS systems, applications, or database. You can configure multiple Multi-Protocol Gateways, one for different endpoint configuration or message policy configuration.

A Multi-Protocol Gateway has the following capabilities:

- ▶ Implements Reliable Messaging policies.
- ▶ Implements the WS-Addressing protocol enforcement.
- ▶ Accepts and sends SOAP, raw XML, or unprocessed (binary) documents.
- ▶ Transforms XML to binary format documents and binary format documents to XML.
- ▶ Filters, validates, transforms, encrypts, or decrypts XML documents.
- ▶ Routes XML documents.
- ▶ Signs documents or verifies signatures.
- ▶ Processes large documents in the streaming mode.
- ▶ Implements message-level security or service-level security, including WS-Security, WS-Trust, SAML, and LDAP.
- ▶ Communicates with clients, servers, and peers with SSL encryption.
- ▶ Monitors and controls data traffic based on request sources and requested resources.
- ▶ Allows, rejects, strips, or processes attachments (MIME, DIME, and MTOM).

12.2.11 WS-Proxy

WebSphere DataPower supports schema validation, policy application, SLA monitoring, load-balancing with peers based on SLA, UDDI directing, governance tie-in, and more.

12.2.12 XML management interface

This interface, which is known as SOAP Management Interface (SOMA), accepts SOAP request messages that are sent from any client. These requests are customized according to the SOMA schema and WSDL inside WebSphere DataPower, and are sent to WebSphere DataPower through port 5550 to manage, create, and delete all objects as necessary. A common usage of the XML Management Interface is for backups and the status of the device.

12.2.13 XML manager

The XML manager controls the parsing of every message that comes through WebSphere DataPower. An XML Manager obtains and manages XML documents, style sheets, and other document resources on behalf of one or more services. It also can provide the following functions:

- ▶ Set manager-associated limits on the parsing of XML documents
- ▶ Enable and set levels for document caching
- ▶ Perform extension function mapping
- ▶ Enable XML-manager-based schema validation
- ▶ Schedule an XML-manager-initiated Processing Rule
- ▶ Contains the User Manager Objects settings
- ▶ Domain backups weekly
- ▶ FullSys backups quarterly
- ▶ Manual domain backups sporadically upon handing off service level objects from DEV to the acceptance domain

12.2.14 XML processing

WebSphere DataPower can speed up XML processing by offloading it from servers and networks. It can perform XML parsing, XML schema validation, XPath routing, XSLT, XML compression, and other essential XML processing with wirespeed XML performance. WebSphere DataPower also provides real-time XML statistics, such as throughput, traffic statistics, errors, and other processing statistics.

12.2.15 XSL co-processor

WebSphere DataPower provides accelerated XML parsing, schema validation, and XSLT processing.

12.3 WebSphere DataPower and IMS integration solutions

The IBM WebSphere DataPower Integration Appliance provides three types of support for IMS:

- ▶ Inbound access to IMS transactions from an external client
Access to IMS TM through WebSphere DataPower allows an external application to initiate a transaction request to an application program that is running in an IMS TM dependent region and fetch data back
- ▶ Outbound support from IMS synchronous callout to an external client
Allows synchronous callout requests from application programs running in IMS TM systems to data or service providers running on the WebSphere DataPower back end. This is also referred to as an IMS consumer scenario.
- ▶ Access to IMS databases
Access to IMS DB allows an external application to issue SQL calls against IMS databases using the IMS Universal JDBC driver that is delivered with WebSphere DataPower.

The following sections describe these three types of support for IMS.

12.3.1 Requirements for inbound access to IMS transactions

For inbound access to IMS transactions, you must meet the prerequisites that are described in the following sections.

Software requirements

- ▶ IMS V11 or later.
- ▶ A supported WebSphere DataPower firmware release. Check the IBM Support Portal for the latest supported firmware versions and recommended upgrade levels for WebSphere DataPower SOA Appliances at the following website:
<http://www.ibm.com/support/docview.wss?uid=swg21237631>
- ▶ If data transformation is required, use a data map or style sheet. Use the recommended IBM WebSphere Transformation Extender Design Studio to create data transformation maps, or you can code style sheets yourself.

Hardware requirements

IBM WebSphere DataPower Appliance XI50, XI52, XI50B, XI50Z, or XB62

12.3.2 Requirements for IMS Synchronous Callout support

For inbound access to IIMS Synchronous Callout, you must meet the prerequisites that are described in the following sections.

Software requirements

- ▶ IMS V12 with PTF UK82636 and IMS Connect PTF UK91544 or later
- ▶ IBM WebSphere DataPower Firmware V6.0 or later
- ▶ If data transformation is required, use a data map or style sheet. Use the recommended IBM WebSphere Transformation Extender Design Studio to create data transformation maps, or you can code style sheets yourself.

Hardware requirements

IBM WebSphere DataPower appliance XI52, XI50B, or XB62

12.3.3 Requirements for access to the IMS database

For inbound access to IIMS database, you must meet the prerequisites that are described in the following sections.

Software requirements

- ▶ IMS V12 or higher, with the following IMS components enabled:
 - IMS Catalog
 - The Open Database (ODBM) component of the IMS Common Service Layer (CSL)
 - The Structured Call Interface (SCI) component of CSL
- ▶ IBM WebSphere DataPower Firmware V6.0 or later

Hardware requirements

IBM WebSphere DataPower appliance XG45, XI52, XI50B, or XB62

12.3.4 Inbound access to IMS transactions from an external client

WebSphere DataPower provides a back side handler (an IMS Connect object) for communicating with an IMS application. The IMS Connect object handles IMS protocol communications from a WebSphere DataPower service to IMS applications. The configuration of the IMS Connect object defines the behavior of the connection to IMS TM.

Figure 12-2 shows a WebSphere DataPower configuration that supports IMS inbound requests.

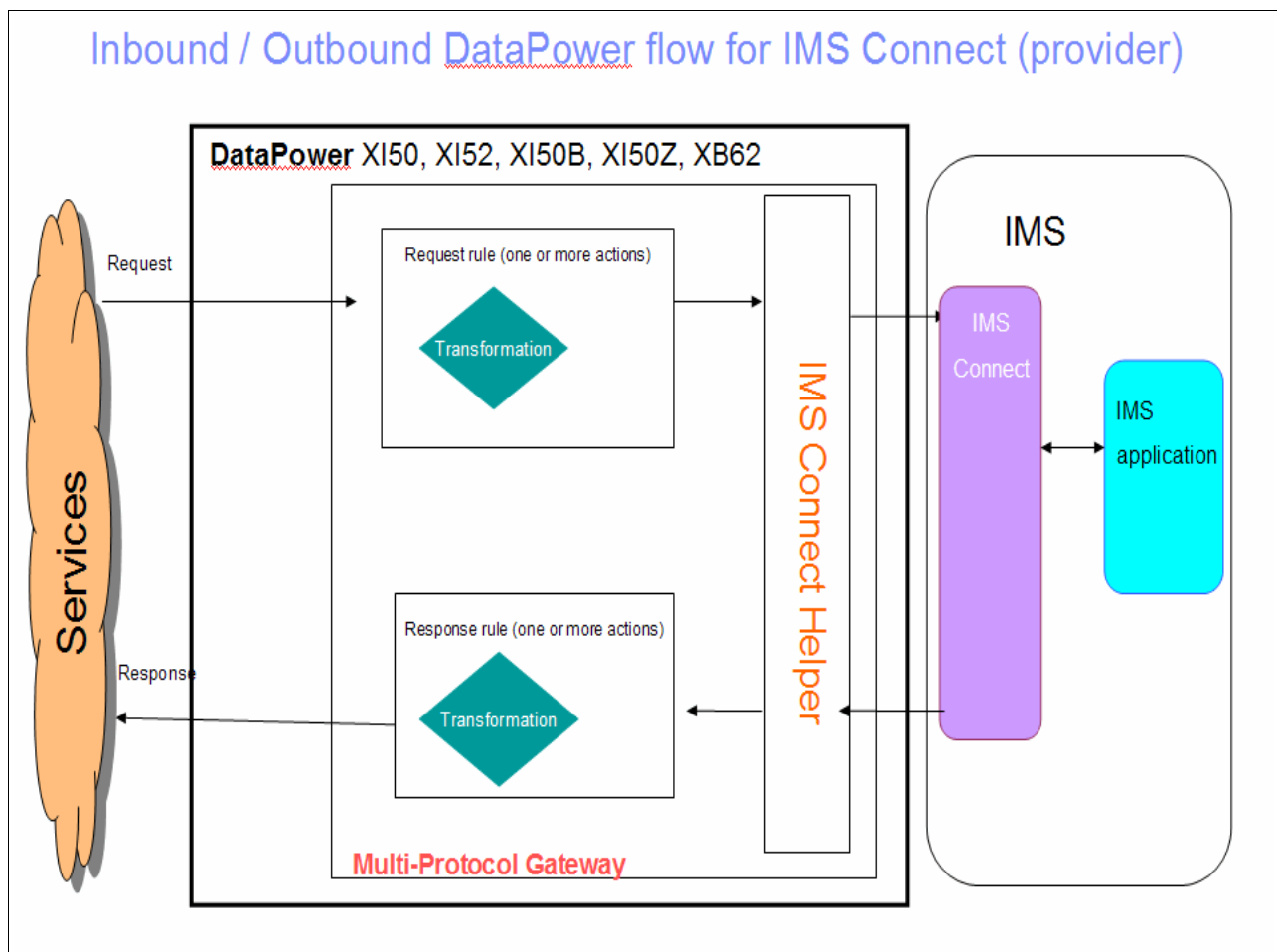


Figure 12-2 Inbound data flow to IMS as a service provider

To configure WebSphere DataPower and IMS for inbound access to an IMS application, you must configure components in both the IMS and WebSphere DataPower environments. In the IMS environment, you need to perform the following actions:

- ▶ Enable OTMA.
- ▶ Configure IMS Connect.

Enabling OTMA

To enable IMS to use OTMA, specify the z/OS cross-system coupling facility (XCF) group name and IMS OTMA member name during system definition.

OTMA is installed with IMS TM. The IMS INSTALL/IVP dialog is not used to install OTMA.

To start OTMA, you can use the OTMA=Y start parameter in the IMS procedure during IMS system definition or, after an IMS restart, issue the type-1 command **/START OTMA**.

Configuring IMS Connect

IMS Connect **HWS** and **TCPIP** statements are required for all types of IMS Connect support. All of the IMS Connect configuration statements are defined in the **HWSCFGxxx** member of the **IMS.PROCLIB** data set.

The port number on which IMS Connect listens for WebSphere DataPower is defined in the **TCPIP** configuration statement with the **PORT** or **PORTID** keyword.

In the **TCPIP** statement, you must also specify the exit routine that communicates with WebSphere DataPower by using the **EXIT=** parameter.

Configuring WebSphere DataPower for IMS inbound requests

To configure WebSphere DataPower to support inbound requests to IMS from an external data or service, complete the following steps:

1. Configure the WebSphere DataPower Multi-Protocol Gateway.
2. Configure the WebSphere DataPower Front Side Protocol handler for incoming request into WebSphere DataPower.
3. Define the WebSphere DataPower Multi-Protocol Gateway Processing Policy that determines the actions that WebSphere DataPower takes with the inbound requests and responses that it handles.
4. Configure the IMS Connect object for back side setting.
5. Apply the changes and save the configuration.

These steps are described in more detail in the following sections.

Configuring the WebSphere DataPower Multi-Protocol Gateway

To configure the Multi-Protocol Gateway, complete the following steps:

1. In the WebSphere DataPower Control window, select **Multi-protocol Gateway**, as shown in Figure 12-3, and click **Add**.

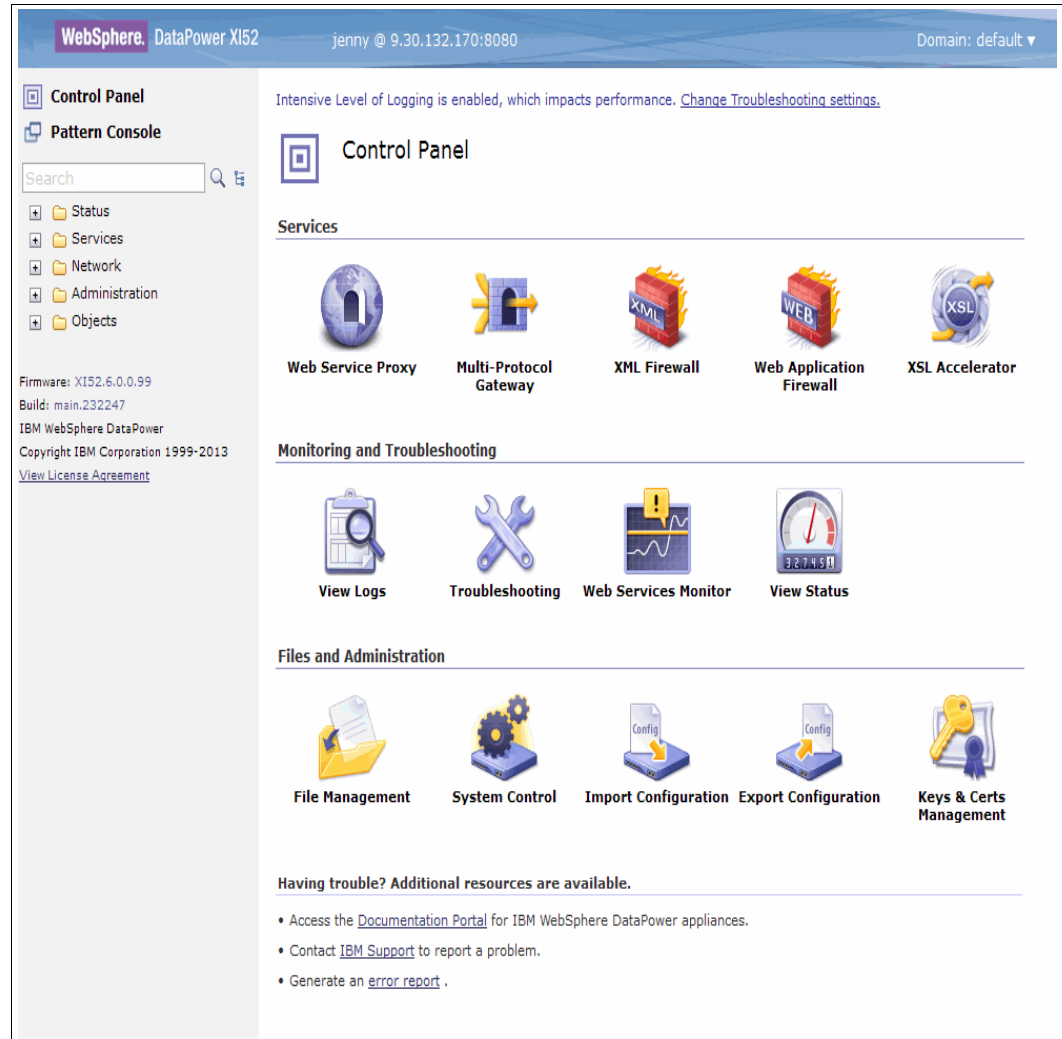


Figure 12-3 WebSphere DataPower Control Panel

2. In the Configure Multi-Protocol Gateway window, specify a Multi-Protocol Gateway name, as shown in Figure 12-4.

WebSphere DataPower XI52 jenny @ 9.30.132.170:8080 Domain: default Save Config

Control Panel Pattern Console

Search

☐ Status
☐ Services
☐ Network
☐ Administration
☐ Objects

Firmware: XI52.6.0.0.99
 Build: main.232247
 IBM WebSphere DataPower
 Copyright IBM Corporation 1999-2013
[View License Agreement](#)

Intensive Level of Logging is enabled, which impacts performance. [Change Troubleshooting settings.](#)

Configure Multi-Protocol Gateway

General Advanced Subscriptions Policy SLA Policy Details Stylesheet Params Headers Monitors WS-A

Apply Cancel Help

General Configuration

Multi-Protocol Gateway Name *

Summary

Type
☐ dynamic-backends
☒ static-backend *

XML Manager
 default + ... *

Multi-Protocol Gateway Policy
 (none) + ... *

URL Rewrite Policy
 (none) + ...

Back side settings

Default Backend URL *

MQ Helper Tibco EMS Helper
 WebSphere JMS Helper IMSConnect Helper

Front side settings

Front Side Protocol
 (empty)
 Add + ... *

User Agent settings

Match	Property
<small>Note: To edit the User Agent, please access via the XML Manager above.</small>	

SSL Client Crypto Profile
 (none) + ...

Response Type

☐ JSON
☐ Non-XML
☐ Pass through
☒ SOAP
☐ XML

Request Type

☐ JSON
☐ Non-XML
☐ Pass through
☒ SOAP
☐ XML

Figure 12-4 WebSphere DataPower Multi-Protocol Gateway configuration

3. Towards the bottom of the panel, you must set the request type and response type. The request type defines the traffic between IMS and WebSphere DataPower on the front end. The response type defines the traffic between the service provider and WebSphere DataPower on the back end. For example:
 - To configure the gateway to process IMS synchronous callout requests, you must select **Non-XML** for both request and response types.
 - To configure the gateway to process request with JSON data, you must specify **JSON** for both request and response types.

More advanced settings, such as timeout values for front- and back-side connections, are available under the Advanced tab, as shown in Figure 12-5.

The screenshot displays the 'Configure Multi-Protocol Gateway' interface in the WebSphere DataPower XI52 control panel. The 'Advanced' tab is selected, showing a variety of configuration options. On the left, a sidebar contains a 'Control Panel' and 'Pattern Console' section with a search bar and a tree view of system components like Status, Services, Network, Administration, and Objects. Below this, firmware and build information are listed. The main content area is titled 'Configure Multi-Protocol Gateway' and includes tabs for General, Advanced, Subscriptions, Policy, SLA Policy Details, Stylesheet Params, Headers, Monitors, and WS-A. The 'Advanced' tab is active, revealing settings for persistent connections, timeouts, and message processing. Key settings include 'Persistent Connections' (on), 'Allow Cache-Control Header' (on), 'Loop Detection' (on), 'Follow Redirects' (on), 'Error Policy' (none), 'Allow Chunked Uploads' (on), 'Process Backend Errors' (on), 'Proxy HTTP Response' (on), 'Front Persistent Timeout' (180 seconds), 'Back Persistent Timeout' (180 seconds), 'Monitor via Web Services Management Agent' (on), 'Message capture via Web Services Management Agent' (All), 'Advanced Crypto' (Gateway Credentials: none), 'MIME Back Header Processing' (on), 'MIME Front Header Processing' (on), 'Service Priority' (Normal), 'Default Param Namespace' (http://www.datapower.com/param/config), 'Query Param Namespace' (http://www.datapower.com/param/query), 'SOAP Schema URL' (store:///schemas/soap-envelope.x), 'Load Balancer Hash Header' (empty), 'Message Processing Modes' (Request rule in order, Backend in order, Response rule in order), and 'Process Messages Whose Body Is Empty' (on).

Figure 12-5 Advanced configuration in Multi-Protocol Gateway

WebSphere DataPower Front Side Protocol Handler

Depending on the model and version of the device, WebSphere DataPower offers a selection of front-side protocol handlers. The front-side handler configuration defines the source from which input requests come into WebSphere DataPower. It determines specific network communication protocol, address, port, and other protocol-specific settings.

A common front-side handler is the HTTPS Front Side Handler. The handler receives HTTP requests sent to the device over SSL and forwards them to the appropriate gateway.

You can add a specific front side protocol handler under Front side settings, as shown in Figure 12-6.

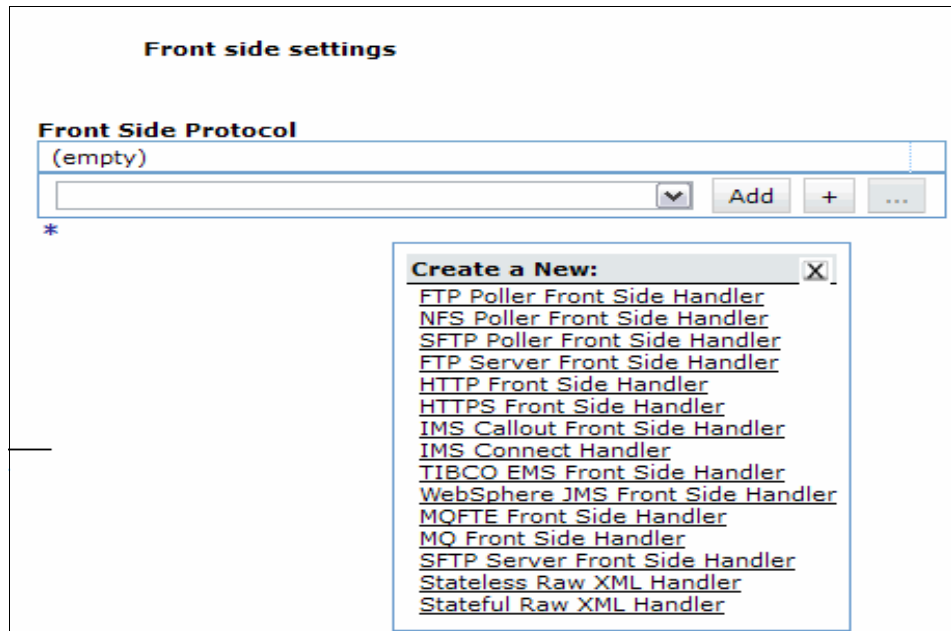


Figure 12-6 Adding a front-side handler to the Multi-Protocol Gateway

WebSphere DataPower Multi-Protocol Gateway processing policy

A Multi-Protocol Gateway processing policy defines the actions WebSphere DataPower takes against data that is passed through to the endpoint service. It consists of one or more rules. Each rule can be applied to data on client/server, server/client, both directions, or error direction.

A rule is depicted as a line in the policy definition window, and consists of the actions shown in Figure 12-7.

Configure Multi-Protocol Gateway Style Policy

Policy:

Policy Name: *

Apply Policy Cancel

Rule:

Rule Name: Rule Direction: Both Directions ▼

New Rule Delete Rule

Create rule: Click New, drag action icons onto line. Edit rule: Click on rule, double-click on action.

Filter Sign Verify Validate Encrypt Decrypt Transform Route AAA Results Advanced

ORIGIN SERVER CLIENT

Create Reusable Rule

Configured Rules			
Order	Rule Name	Direction	Actions
There are no rules defined for this policy.			

Figure 12-7 Configuring a Multi-Protocol Gateway Policy with rules and actions

A user can assemble a rule by dragging actions on to the line.

Here are some common actions:

- ▶ Match action: Specifies the matching criteria for the rule. To configure a match rule, you can choose from the match templates that are provided, for example, URL match template, HTTP header tag match template, error code match template, XPath match template, and HTTP method template.
- ▶ Filter action: Specifies an XSL style sheet to use for the document filter.
- ▶ AAA action: Specifies the authentication and authorization policy.
- ▶ Transform action: Specifies data transformation with XSL style sheet and WebSphere Transformation Extender maps. An XSL style sheet can be used to map and direct the requests to back-end points, setting data values, selecting the WebSphere Transformation Extender maps to invoke, and so on. A WebSphere Transformation Extender map can be used to transform binary data (data type COBOL or PL/I) to XML and vice versa. For more information about creating data transformation maps and type trees, see the WebSphere Transformation Extender documentation in the WebSphere Transformation Extender information center at the following website:
<http://pic.dhe.ibm.com/infocenter/wtxdoc/v8r4m1/index.jsp>
- ▶ Results action: Specifies the handling of data to deliver to the endpoints, for example, asynchronous, number of retries, retry interval, HTTP method, and destination context.

Configuring an IMS Connect object for a back-side setting

To enable a Multi-Protocol Gateway to send incoming requests (SEND-RECEIVE only) to IMS Connect, you must configure the back-side setting by clicking **IMSConnect Helper** and creating an IMS Connect URL, as shown in Figure 12-8.

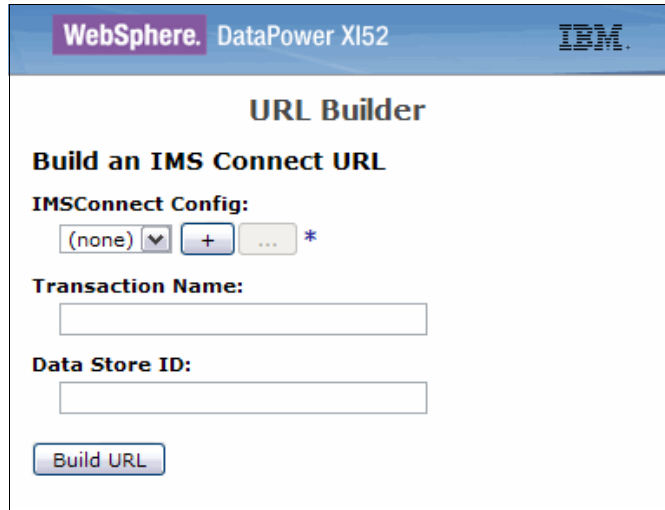
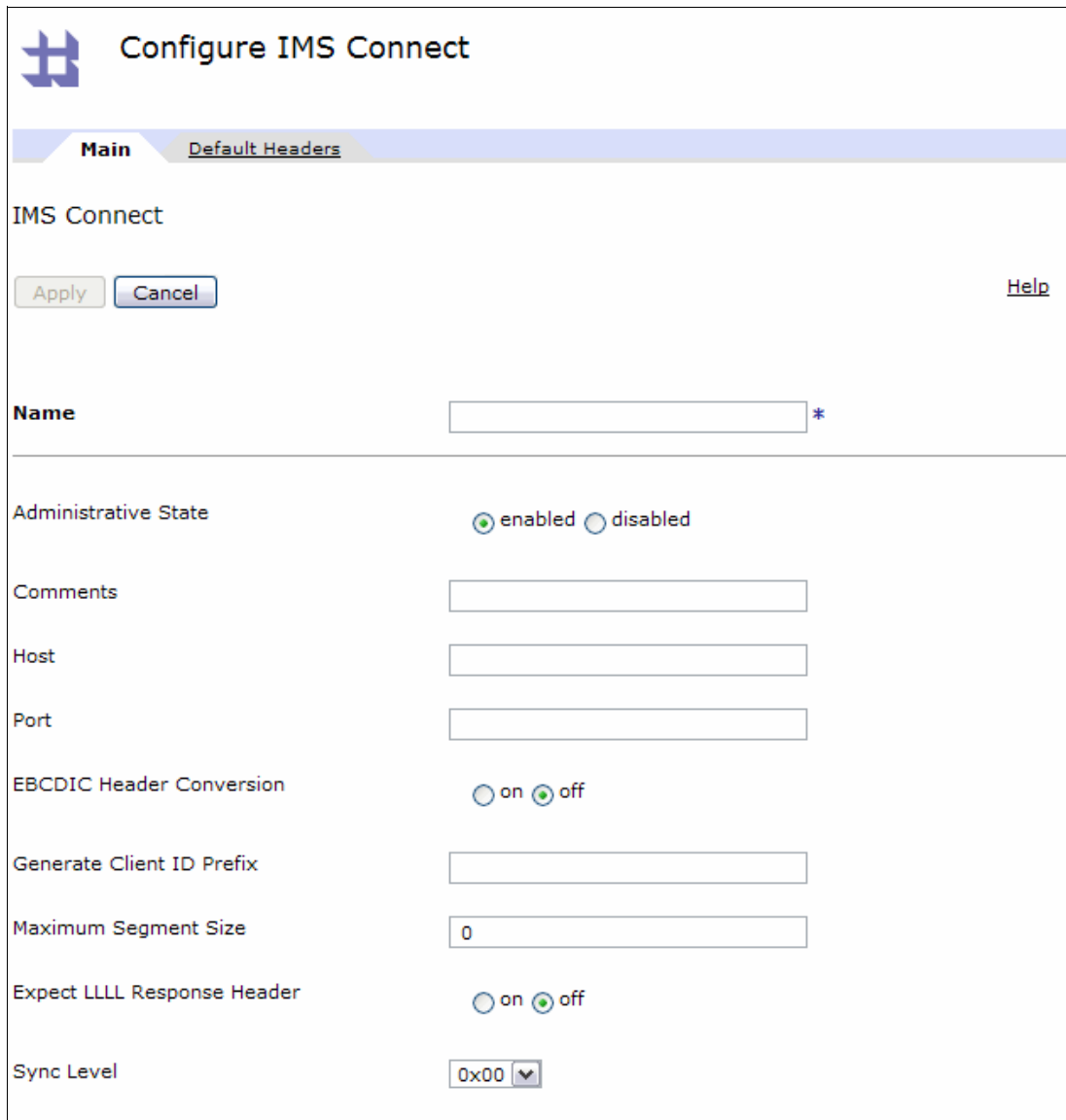
The screenshot shows the 'URL Builder' interface within the WebSphere DataPower XI52 console. The header bar includes 'WebSphere. DataPower XI52' and the IBM logo. The main title is 'URL Builder'. Below it, the instruction 'Build an IMS Connect URL' is displayed. The 'IMSConnect Config:' section features a dropdown menu currently set to '(none)', a '+' button, a button with three dots, and an asterisk. Below this are two text input fields: 'Transaction Name:' and 'Data Store ID:'. At the bottom of the form is a 'Build URL' button.

Figure 12-8 Creating an IMS Connect configuration

The main menu is where you configure the Host, Port, Conversion, and Client_ID prefix information for the WebSphere DataPower IMS Connect object, as shown in Figure 12-9 on page 411.



The image shows a 'Configure IMS Connect' dialog box. At the top left is a blue icon of four squares. The title is 'Configure IMS Connect'. Below the title are two tabs: 'Main' (selected) and 'Default Headers'. The 'Main' tab contains the following fields and controls:

- IMS Connect**: A section header.
- Buttons**: 'Apply' and 'Cancel' buttons on the left, and a 'Help' link on the right.
- Name**: A text input field with an asterisk (*) indicating it is required.
- Administrative State**: Two radio buttons, 'enabled' (selected) and 'disabled'.
- Comments**: A text input field.
- Host**: A text input field.
- Port**: A text input field.
- EBCDIC Header Conversion**: Two radio buttons, 'on' and 'off' (selected).
- Generate Client ID Prefix**: A text input field.
- Maximum Segment Size**: A text input field containing the value '0'.
- Expect LLLL Response Header**: Two radio buttons, 'on' and 'off' (selected).
- Sync Level**: A dropdown menu showing '0x00'.

Figure 12-9 Defining information for the IMS Connect object

- ▶ **Host**: Specify the host name or IP address of the IMS TCP/IP server and IMS Connect.
- ▶ **Port**: Specify the port on which the IMS TCP/IP server is running.
- ▶ **EBCDIC header conversion**: This option can be turned on for converting the headers to EBCDIC. The IMS Connect user message exit can process EBCDIC data. Some IMS Connect exits can handle both UTF-8 and EBCDIC. This conversion affects only the headers. Use transformation to do any data conversion in the policy.
- ▶ **Generate client ID prefix**: A two-letter prefix for the generated client ID. DP is used if the prefix is not specified.

Here are other parameters that must be customized:

Exit Program	The IMS Connect user message exit routine to use for all the IMS connections.
Client ID	A string of 1 - 8 uppercase alphanumeric (A - Z, 0 - 9) or special (@, #, and \$) characters, left-aligned, and padded with blanks. It specifies the name of the client ID that is used by IMS Connect. If this string is not supplied from the client, then the IMS Connect user message exit routine generates it.
Transaction code	The code of the transaction to invoke in IMS.
Data store	Specifies the Datastore name (IMS destination ID).
Logical terminal name	The LTERM override value to be used by OTMA.
RACF ID	The plain text string sent to the server to identify the client.
RACF Password	The host security password that is used to log in to the IMS TCP/IP server and IMS Connect.
RACF Group	The group the Host security ID belongs to.
Encoding scheme	Select the Unicode encoding schema. Leave as (none) so that it is set dynamically in the IMS header.
IRM Timer	Specifies the amount of time that IMS Connect waits for IMS to return response data. An example value of 21 is set to an IRM Timer value of 0.21 sec. For more information about specifying timeout values, see IMS Connect timeout specifications in the IMS documentation at http://pic.dhe.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.ims12.doc.ccg/ims_ct_timeout_specs.htm .

After applying the changes, the confirmation window opens. On the Multi-Protocol Gateway window, click **Apply** again. Finally, click **Save config**.

On the View Status window, the Object Status of the recently added IMS Connect Object is displayed. The following statuses are possible:

- ▶ Invalid: Invalid Configuration
- ▶ Saved: Persisted Configuration
- ▶ New: New Configuration
- ▶ Modified: Modified Configuration
- ▶ Deleted: Deleted Configuration
- ▶ External: External Configuration

12.3.5 Outbound support from IMS synchronous callout to an external client

WebSphere DataPower provides a front-side handler, IMS Callout Front Side Handler, for retrieving ICAL requests from an IMS application. The IMS Callout Front Side handler of the WebSphere DataPower Multi-Protocol Gateway processes IMS callout requests and responses with an external back-end client application.

Figure 12-10 shows the data flow for an IMS synchronous callout request to the external service and response to IMS through WebSphere DataPower.

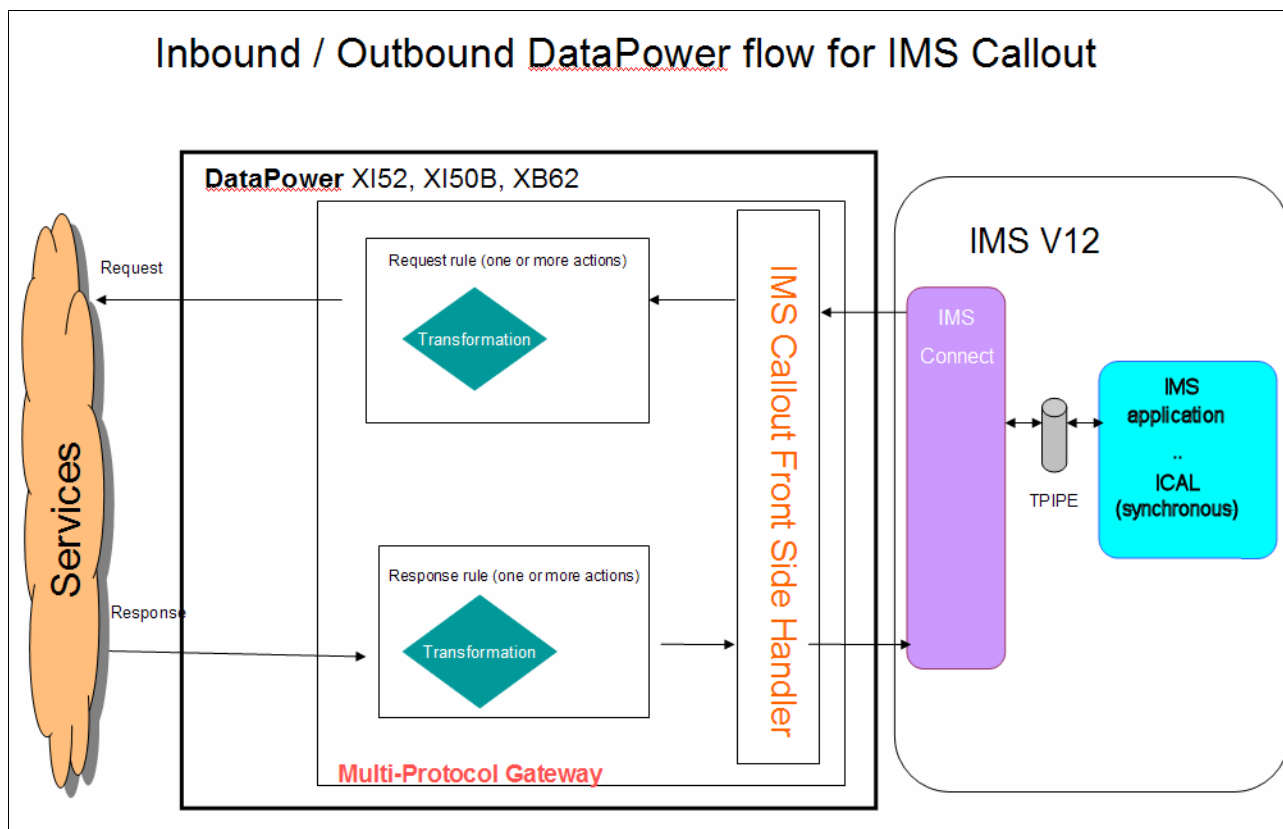


Figure 12-10 Outbound data flow from IMS as service consumer

To configure WebSphere DataPower and IMS to support access to the IMS TM server, you must configure components in both the IMS and WebSphere DataPower environments:

In the IMS environment, you must complete the following steps:

1. Configure OTMA for synchronous callout support.
2. Configure IMS Connect for synchronous callout support.
3. Code the ICAL call for synchronous callout requests.

Configuring OTMA for synchronous callout support

To support synchronous callout, OTMA must be enabled in IMS and an OTMA destination descriptor must be defined that routes the callout requests through IMS Connect and the WebSphere DataPower appliance.

Enabling OTMA

To enable IMS to use OTMA, specify the z/OS cross-system coupling facility (XCF) group name and IMS OTMA member name during system definition.

OTMA is installed with IMS TM. The IMS INSTALL/IVP dialog is not used to install OTMA.

To start OTMA, you can use the **OTMA=Y** startup parameter in the IMS procedure during IMS system definition or, after an IMS restart, issue the type-1 command **/START OTMA**.

Defining an OTMA destination descriptor for synchronous callout support

An OTMA destination descriptor defines an output destination, or TPIPE, for IMS output messages, such as synchronous callout messages. The WebSphere DataPower IMS Callout Front Side Handler retrieves synchronous callout messages from IMS by listening on the TPIPE that is specified in the OTMA destination descriptor.

You can also use the OTMA destination descriptor to specify a timeout value for synchronous callout requests. If a timeout value is specified in both the OTMA destination descriptor and in the DL/I ICAL call itself, the lesser of the two values is used.

OTMA destination descriptors can be created, modified, or deleted while IMS is running by using IMS type-2 commands, or they can be coded during IMS system definition and stored in the DFSYDTx member of the IMS.PROCLIB data set. However, IMS must be restarted to recognize any new or changed OTMA destination descriptors that are coded in the DFSYDTx member.

Here is an example of an OTMA destination descriptor:

```
D OTMDSC01 TYPE=IMSCON TMEMBER=HWS1 TPIPE=TPIPE1
D OTMDSC02 TYPE=IMSCON TMEMBER=HWS1 TPIPE=TPIPE2
D OTMDSC03 TYPE=IMSCON TMEMBER=HWS1 TPIPE=TPIPE3
```

Note: The TPIPEs must be dedicated to WebSphere DataPower and the synchronous callout requests must be sent to a particular service. The TPIPEs cannot be shared with any other application or solution, such as IMS SOAP Gateway. If a TPIPE is shared, either WebSphere DataPower or the other solution might be unable to retrieve the synchronous callout requests correctly.

For more information about OTMA destination descriptors, see the “OTMA destination descriptors” topic at the following website:

http://pic.dhe.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.ims12.doc.ccg/ims_otma_admin_006.htm

For more information about the **CREATE OTMADESC** command and its keywords, see the “CREATE OTMADESC command” topic at the following website:

http://pic.dhe.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.ims12.doc.cr/imscmds/ims_createotmadesc.htm#ims_createotmadesc

For more information about coding OTMA destination descriptors, see the “OTMA destination descriptor syntax and parameters” topic at the following website:

http://pic.dhe.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.ims12.doc.sdg/ims_dfsydtx_proclib_dest_dscrp.htm

Configuring IMS Connect for synchronous callout support

IMS Connect must be configured for IMS TM access with the following tools:

- ▶ A **DATASTORE** configuration statement
- ▶ The IBM WebSphere DataPower message exit routine (HWSDPWR1)

An IMS Connect **DATASTORE** statement defines a connection between IMS Connect and IMS TM. It is required for synchronous callout support in addition to the **HWS** and **TCPIP** statements that are required for all types of IMS Connect support. All of the IMS Connect configuration statements are defined in the HWSCFGxxx member of the IMS.PROCLIB data set.

The value of the ID keyword in the **DATASTORE** statement is the value that is specified in WebSphere DataPower in the Data store field when you configure the IMS Callout Front Side Handler.

The port number on which IMS Connect listens for WebSphere DataPower is defined in the **TCPIP** configuration statement with the **PORT** or **PORTID** keyword.

In the **TCPIP** statement, you must also specify the HWSDPWR1 exit routine on the **EXIT=** parameter. The HWSDPWR1 exit routine was added to IMS Version 12 by PTF UK91544 and is available as object code only, so it is not customizable.

To make the HWSDPWR1 exit routine available to IMS Connect, you must link the exit routine by using a job that is similar to the one that is shown in Example 12-1.

Example 12-1 Linking the HWSDPWR1 exit routine

```
//HWSDPWR1 JOB LINK,MSGLEVEL=1,REGION=640K,CLASS=G
//*-----*
//* Link the exit                               *
//*-----*                                     //LINKMOD
EXEC PGM=IEWL,

//          PARM='SIZE=(180K,28K),RENT,REFR,NCAL,LET,XREF,LIST,TEST'
//SYSPRINT DD SYSOUT=A
//SYSLMOD  DD UNIT=SYSVIO,DISP=(,PASS),SPACE=(TRK,(1,1,1)),
//          DSN=&&CSDM17
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//SYSLIN   DD DSN=IMSDTPWR.OBJECT.LIB             <== User defined info
//LINK1    EXEC PGM=IEWL,
//          PARM=('SIZE=(880K,64K)',RENT,REFR,
//          NCAL,LET,XREF,LIST,TEST)
//SYSPRINT DD SYSOUT=A
//TEXT     DD UNIT=SYSVIO,DISP=(OLD,PASS),DSN=&&CSDM17
//SYSLMOD  DD DSN=ICONEXIT.OBJECT.LIB,             <== Exit Object lib
//          DISP=SHR,UNIT=SYSDA,VOL=SER=SDV000
//RESLIB   DD DSN=IMSB LD.I12STSM.M.CRESLIB,DISP=SHR <== IMS RESLIB
//SYSUT1   DD UNIT=SYSVIO,DISP=(,DELETE),SPACE=(CYL,(10,1),RLSE)
//SYSLIN   DD *
//          INCLUDE TEXT(HWSDPWR1)
//          ENTRY HWSDPWR1
//          MODE RMODE(31),AMODE(31)
//          NAME HWSDPWR1(R)
//
```

Coding the ICAL call for synchronous callout requests

The ICAL call of the IMS DL/I API is how an application program running in an IMS TM dependent region makes a synchronous callout request to a data or service provider on the WebSphere DataPower back end.

You must code the application programs to build and issue the ICAL call. The fields of the ICAL call are defined by an application interface block (AIB). In the AIB fields, the application program specifies the attributes and content of the callout request, including the following items:

- ▶ The name of the OTMA destination descriptor.
- ▶ Optionally, a timeout value in one-hundredths of a second.

- ▶ The length of the request data.
- ▶ The length of the response data.
- ▶ A 1 - 8 byte mapname as the first 8 bytes in AIBUTKN so that this ID can be included in the OTMA state data in the callout message. The ID can be used as a unique service identifier for data transformation mapping and service routing

When a timeout value for a synchronous callout request is specified in both the OTMA destination descriptor and in the DL/I ICAL call itself, IMS uses the lower of the two values.

Synchronous callout messages that are sent from IMS by using the ICAL call do not use the IMS message queues. Consequently, synchronous callout messages are not constrained to the 32 KB message segment restriction that is imposed by the IMS message queue.

For a description of the parameter fields of an ICAL call and the valid values, see the “ICAL call” topic at the following website:

http://pic.dhe.ibm.com/infocenter/dzichelp/v2r2/topic/com.ibm.ims12.doc.apr/ims_icalcalltm.htm

Configuring WebSphere DataPower for IMS synchronous callout requests

To configure WebSphere DataPower to support synchronous callout requests from IMS to a data or service that is provided on the WebSphere DataPower back side, complete the following steps:

1. Configure the DataPower Multi-Protocol Gateway.
2. Configure the IMS Callout Front Side Handler.
3. Define the Multi-Protocol Gateway processing policy for IMS Callout that determines the actions that WebSphere DataPower takes on the callout requests and responses that it handles.
4. Configure the Multi-Protocol Gateway back-end setting to deliver the callout request and to wait for the response.
5. Apply the changes and save the configuration.

IMS Callout Front Side Handler


To enable a Multi-Protocol Gateway to retrieve IMS ICAL callout requests from IMS, you must add an IMS Callout Front Side Handler. The IMS Callout Front Side Handler also manages the return of the response data to the IMS application.

By default, IMS Callout Front Side Handlers are enabled when they are created. During the configuration of a Multi-Protocol Gateway, disable the IMS Callout Front Side Handler until the Multi-Protocol Gateway and the back-end service is ready to service requests. Otherwise, any callout requests sent to the IMS Callout Front Side Handler produce an error.

Also, if an IMS Callout Front Side Handler is enabled while the Multi-Protocol Gateway is being configured, each time you apply changes by clicking **Apply**, the Multi-Protocol Gateway effectively restarts the IMS Callout Front Side Handler and performs tear down and resume tpipe operations. When the Multi-Protocol Gateway is ready, you can enable the IMS Callout Front Side Handler to start the retrieval of IMS ICAL requests.

You can configure one or more IMS Callout Front Side Handlers in a single Multi-Protocol Gateway. For each front-side protocol handler for IMS Callout support, you can configure one or more TPIPEs on the same IMS Connect host and port, and the same IMS data store.

To configure the IMS Callout Front Side Handler, specify the following IMS system parameters. The required fields have an asterisk next to them, as shown in Figure 12-11.



Configure IMS Callout Front Side Handler

Main

Advanced

IMS Callout Front Side Handler

Apply

Cancel

Help

Name

*

Administrative State

enabled

disabled

Comments

RemoteDevelop

on

off

RemoteDevelopServer

Host

*

Port

*

Data store

*

OTMA tppe names

(empty)

add

*

SAF user name

SAF password

SAF group

Retry attempts

5

Retry interval

3

seconds

Figure 12-11 Configuring IMS Callout Front Side Handler

- Host

Specify the host name or IP address of the target IMS Connect server.
- Port

Specify the port on which the IMS TCP/IP server, IMS Connect, is listening for WebSphere DataPower.

Data store	Specify the name of the IMS data store. The value that is specified here must match the value that is specified on the ID keyword of an IMS Connect DATASTORE configuration statement.
OTMA tpipe names	Specifies the IMS OTMA tpipe names. WebSphere DataPower passes the TPIPE name to IMS Connect as an alternative client ID. The TPIPE names that are defined in this window must match a tpipe name that is specified in an IMS OTMA destination descriptor in the DFSYDTx member of the IMS.PROCLIB data set.

Note: Do not specify a TPIPE that is used for anything other than synchronous callout requests for a particular service provider. TPIPEs cannot be shared by any other application or solution. If a TPIPE is shared, either WebSphere DataPower or the other solution might be unable to retrieve the synchronous callout requests correctly.

SAF user name	Specify the security authorization facility (SAF) user name. The value can be up to eight characters in length and cannot be blank. The value can use all alphanumeric characters and the following special characters: @, #, and \$.
SAF password	Specify the security authorization facility (SAF) password. The value can use all alphanumeric characters and the following special characters: @, #, and \$.
SAF group	Specify the name of the security authorization facility (SAF) group. The value can be up to eight characters in length and cannot be blank. The value can use all alphanumeric characters and the following special characters: @, #, and \$.
Retry attempts	Specify the number of times to attempt to resume a transaction pipe (tpipe) after processing encounters an error. Enter a value in the range 1 - 256. The default value is 5.
Retry interval	Specify the number of seconds to wait before processing attempts to resume the transaction pipe (tpipe). The minimum value is 1. The default value is 3.

More advanced settings, such as tracing and connection timeout values, are available in the Advanced tab.

Trace file

Enables IMS tracing and specifies the location of the trace file. You can use the following directories as the location when you enable tracing:

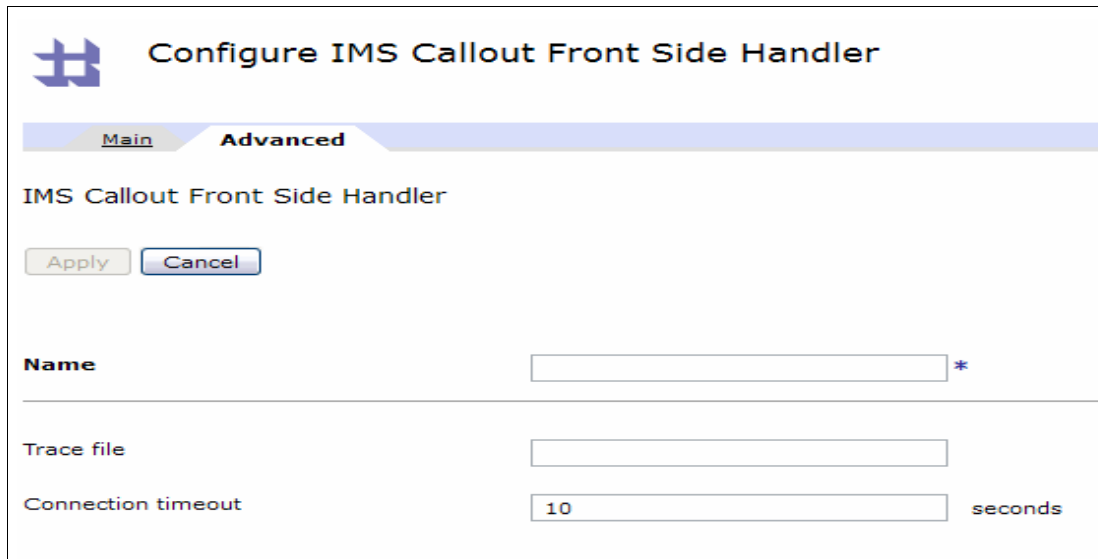
- ▶ logtemp
- ▶ logstore
- ▶ temporary

Note: Enable tracing when instructed by IBM Support to help with problem determination.

If you enabled tracing and want to disable, remove the directory and apply the configuration.

Connection timeout

Specify the number of seconds that the appliance waits to establish a connection to IMS Connect. A value of 0 disables the timeout. The default value is 10. For more details, see Figure 12-12.



The screenshot shows a web-based configuration interface titled "Configure IMS Callout Front Side Handler". It has two tabs: "Main" and "Advanced", with "Advanced" currently selected. Below the tabs, the title "IMS Callout Front Side Handler" is displayed. There are two buttons: "Apply" and "Cancel". Below these are three configuration fields: "Name" with a text input field and an asterisk, "Trace file" with a text input field, and "Connection timeout" with a text input field containing the value "10" and the unit "seconds".

Figure 12-12 Advanced IMS Callout configuration

After you complete the configuration of the IMS Callout Front Side Handler, click **Apply** on the Front Side Handler window, and the Front Side Handler is enabled. On the Multi-Protocol Gateway window, click **Apply** again. Finally, click **SAVE config**.

The status indicator shows the status of the Front Side Handler:

- | | |
|--------------------------|---|
| [up] | The IMS Callout Front Side Handler is enabled and can communicate with IMS Connect and IMS to actively process RESUME TPIPE . If the IMS Callout Front Side Handler encounters an error, it attempts to process at specified intervals, up to a specified maximum number of attempts before going into [down-pending] status. |
| [down - pending] | The IMS Callout Front Side Handler is enabled but is in recovery mode. The Front Side Handler attempts to ping IMS Connect every 60 seconds to re-establish a connection. It goes in to [up] mode when it gets a successful response from the /DIS OTMA command, which you run to verify that both IMS and IMS Connect are responding. |
| [down - disabled] | The IMS Callout Front Side Handler is disabled. |

For a sample test scenario and its configuration steps, troubleshooting, hints, and tips, see the *WebSphere DataPower IMS Implementation Guide*, found at:

<http://www.ibm.com/support/docview.wss?uid=swg27038927&aid=1>

Multi-Protocol Gateway processing policy for IMS Callout

In addition to the configuration listed in “WebSphere DataPower Multi-Protocol Gateway processing policy” on page 408, this section lists advanced optional tasks that are specific to configuring the transform action for processing IMS Callout request/response.

For a transform action, you can, in the XSL style sheet, optionally perform the following actions:

1. Access the HTTP headers `ims-callout-correlation-token` and `ims-callout-service-id`.

In each IMS Callout request, there are two IMS-specific headers:

- `ims-callout-correlation-token`: A hex representation of the unique ICAL correlation token. This token contains the user ID.
- `ims-callout-service-id`: An 8-byte mapname that is specified in the ICAL AIBUTKN.

For example, in Example 12-2, when an error occurs in the transform action, the IMS correlation token and service ID is written out to the system log with an error level.

Example 12-2 Sample XSL

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  xmlns:dp="http://www.datapower.com/extensions"
  extension-element-prefixes="dp">

  <xsl:template match="/">
    <xsl:variable name="be"
select="dp:request-header('ims-callout-service-id')"/>
    <xsl:choose>
      <xsl:when test="$be = 'SERVICE1'">
        <dp:set-variable name="'var://context/map/name'"
value="'local://request-250-cp037.dpa'" />
        <dp:set-variable name="'var://service/routing-url'"
value="'http://9.55.133.142:6221'" />
      </xsl:when>
      <xsl:when test="$be = 'SERVICE2'">
        <dp:set-variable name="'var://context/map/name'"
value="'local://request-8000-cp037.dpa'" />
        <dp:set-variable name="'var://service/routing-url'"
value="'http://9.55.133.142:6222'" />
      </xsl:when>
      <xsl:otherwise>
        <dp:reject>unknown backend specified</dp:reject>
      </xsl:otherwise>
    </xsl:choose>
    <xsl:message dp:priority="error">
      Correlation token : <xsl:value-of
select="dp:request-header('ims-callout-correlation-token')"/>
    </xsl:message>
    <xsl:message dp:priority="error">
      Service ID : <xsl:value-of
select="dp:request-header('ims-callout-service-id')"/>
    </xsl:message>
  </xsl:template>
</xsl:stylesheet>
```

2. Dynamically direct the request to a back end by setting the `var://service/routing-url` variable.

You can make the back-end URL decision based on the HTTP header `ims-callout-service-id`. For example, in Example 12-2 on page 420, if the header value is `SERVICE1`, the request is sent to the back-end server on port 6221 versus if the header value is `SERVICE2`, where the request is sent to the back-end server on port 6222.

3. Dynamically select a WebSphere Transformation Extender map by setting the `var://context/map/name` variable.

You can make the WebSphere Transformation Extender map selection decision based on the HTTP header `ims-callout-service-id`. For example, in Example 12-2 on page 420, if the header value is `SERVICE1`, use the WebSphere Transformation Extender map `request-250-cp037.dpa` versus if the header value is `SERVICE2`, use the WebSphere Transformation Extender map `request-8000-cp037.dpa`.

Multi-Protocol Gateway back-end setting

In a Multi-Protocol Gateway, you can select two different modes for the back-end setting:

- ▶ Static Backend: The gateway sends all the requests to the specific back-end URL.
- ▶ Dynamic Backend: The gateway determines the URL of the back-end server dynamically during request processing.

In addition, WebSphere DataPower offers multiple back-end protocol objects to choose from for configuring the back-end setting, for example, `IMSCONNECT`, `WebSphere MQ`, and `WebSphere JMS`.

12.3.6 Access to IMS databases

Configuring an IMS Database Connection involves configuring components in both the IMS and WebSphere DataPower environments.

Configuring IMS Components for access to IMS DB

Access to IMS databases from WebSphere DataPower requires configuring the following IMS components:

- ▶ Configuring IMS Connect for access to IMS DB
- ▶ Configuring Open Database Manager (ODBM) for access to IMS DB

OTMA is not used for access to IMS DB.

Configuring IMS Connect for access to IMS DB

An IMS Connect **ODACCESS** statement is required to configure IMS Connect to support access to IMS databases from WebSphere DataPower.

Among other communication attributes, the **ODACCESS** statement defines the port on which IMS Connect listens for database access requests from WebSphere DataPower. The same port number that is specified on the **DRDAPORT** keyword of the **ODACCESS** statement must also be specified in the Port field when the SQL Data Source is configured in WebSphere DataPower.

The **ODACCESS** statement is in addition to the **HWS** and **TCPIP** statements that are required for all types of IMS Connect support. All of the IMS Connect configuration statements are defined in the `HWSCFGxxx` member of the `IMS.PROCLIB` data set.

Configuring Open Database Manager (ODBM) for access to IMS DB

You configure ODBM by specifying an CSLDCxxx member in the IMS.PROCLIB data set.

Among other attributes, the CSLDCxxx member defines an alias name for the IMS DB server where the database is. Optionally, this same alias name can be specified in the `dataStoreName` field on the Advanced tab when the SQL Data Source is configured.

Configuring WebSphere DataPower components for access to IMS DB

Access to the IMS DB database server through WebSphere DataPower requires configuring the following components in the WebSphere DataPower environment:

- ▶ WebSphere DataPower Multi-Protocol Gateway
- ▶ WebSphere DataPower Front Side Protocol Handler
- ▶ WebSphere DataPower SQL data source
- ▶ Multi-Protocol Gateway processing policy to access IMS DB

Note: A back end is not needed to query an IMS Database; WebSphere DataPower classifies such cases as *enrichment scenarios*, involving a call to an external source that is not the intended back end. For our setup, we create a loop feedback using a dynamic back end and XSLT logic.

WebSphere DataPower SQL data source

An IMS database is defined to WebSphere DataPower as an SQL data source. For each IMS database that you access, you must configure a separate SQL data source.

In IMS, a program specification block (PSB) associates an application program with a given database. When you configure the SQL data source in WebSphere DataPower to identify the target database of an application program, you specify the PSB name instead of the database name. The PSB name is specified in the Data Source ID field. An IMS JDBC driver, which comes preinstalled in the WebSphere DataPower appliance, checks the PSB names in the IMS catalog to validate the data source ID.

The SQL data source is used by an SQL action in a processing policy. The SQL action retrieves the data for further processing by the processing policy. Conversely, the processing policy can store the processed data in the configured database instance.

The SQL data source uses the IMS Universal JDBC driver to establish a TCP/IP connection to the IMS system. This allows users to issue dynamic SQL calls to the underlying database, and returns the result set in tabular format.

To define the SQL data source, specify the parameters that are shown in Figure 12-13.

Configure SQL Data Source

main Advanced Data Source Configuration Parameters

SQL Data Source:ECIMS1 [down - Object is disabled]

Apply Cancel Delete Undo Export View Log View St

Administrative State ☐ enabled ☒ disabled

Comments Data Source for IMS1 on hostname

Database Type IMS *

Connection User Name usr001 *

Connection Password *

Data Source ID APOL1 *

Data Source Host ecdata1.vmec.svl.ibm.com *

Data Source Port 9999 *

Limit Returned Data ☐

Maximum Connections 10 *

Figure 12-13 Configuring SQL Data Source

The configuration window allows you to specify the following IMS specific parameters:

Database Type	IMS.
Connection User Name	TSO user name.
Connection Password	TSO password.
Data Source ID	The name of the program specification block (PSB) that identifies the database to connect to.
Data Source Host	The IP address of the host on which the IMS system is.
Data Source Port	The port on which IMS Connect receives database access requests. This port is defined to IMS Connect with the DRDAPORT keyword of the ODACCESS configuration statement.

Accept the defaults for the other parameters.

You can specify additional parameters under the Data Source Configuration Parameters tab as shown in Figure 12-14.

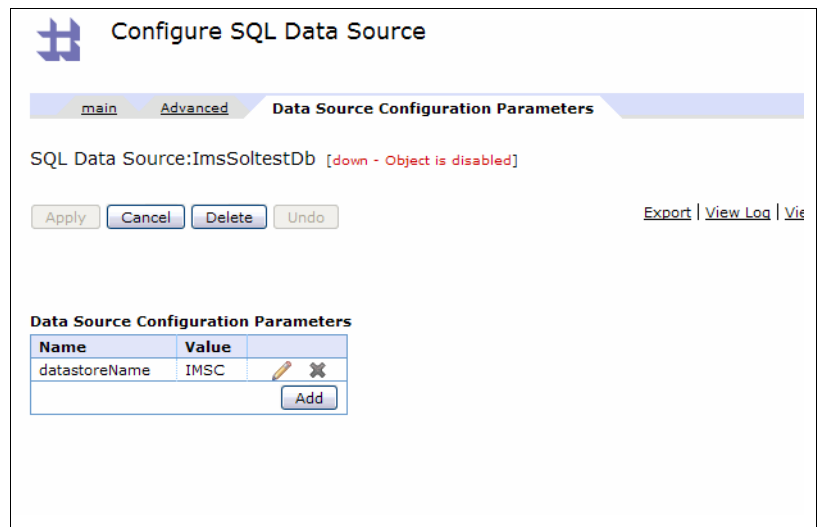


Figure 12-14 Additional parameters for Data Source Configuration

Here are some commonly used name-value pairs:

- datastoreName** The name of your IMS system. This parameter is optional. When the datastoreName is omitted, IMS Connect searches for the appropriate IMS system among the active IMS systems that it is connected to. Specifying the datastoreName can result in a minor performance improvement. When it is specified, the datastoreName value must match an IMS alias name that is defined on the **NAME** keyword of the **ALIAS** statement in the CSLDCxxx member of the IMS.PROCLIB data set.
- traceFile** The full path and file name to which WebSphere DataPower writes log messages. This is used mainly for debugging.
- traceLevel** The level of logging for traceFile. The value -1 indicates that all messages are logged. This is also used for debugging with traceFile.

When you complete the configuration of the SQL data source, click **Apply**. On the Multi-Protocol Gateway window, click **Apply** again.

Multi-Protocol Gateway processing policy to access IMS DB

In addition to the configuration that is described in “WebSphere DataPower Multi-Protocol Gateway processing policy” on page 408, this section lists additional tasks for accessing IMS Database.

Complete the following steps:

1. Transform action: You can specify, in the XSL, to add an extension element to enable an SQL call. An extension element allows parameterized SQL statements. In the element, you must specify the SQL Data Source, PSB name (also specified in the Data Source ID field when the SQL Data Source is configured), and SQL call, as shown in Example 12-3.

Example 12-3 Sample XSL with extension element

```

xsl:stylesheet xmlns:dws="http://ibm.com/datatools/dsws/dataPower"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns:str="http://exslt.org/strings"

```

```

xmlns:regexp="http://exslt.org/regular-expressions"
xmlns:dp="http://www.datapower.com/extensions"
xmlns:date="http://exslt.org/dates-and-times"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0"
extension-element-prefixes="dp" exclude-result-prefixes="dp date stregexp dws">
xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
  <xsl:template match="imsdb"/>
    <xsl:variable name="dbd"/>
      <dp:sql-execute source="'EEnvIMS1'"
        statement="'select * from DFSCAT00.PSB where
          PSB.IOASIZE = 600'"/>
    </xsl:template>
/xsl:stylesheet>

```

2. Advanced Set Variable action: This step skips the back-side processing because a WebSphere DataPower enrichment scenario does not require one:
 - Choose **Set Variable** as the action type.
 - Specify the var://service/mpgw/skip-backside variable name.
 - Set the variable assignment to 1.

Finally, click **Save config**.

For a sample test scenario and its configuration steps, troubleshooting, hints, and tips, see the *WebSphere DataPower IMS Implementation Guide* at the following website:

<http://www.ibm.com/support/docview.wss?uid=swg27038927>

Part 3



Appendixes

This part contains the following appendixes:

- ▶ Appendix A, “Sample code” on page 429
- ▶ Appendix B, “Additional material” on page 461



Sample code

This appendix contains code snippets that are related to items that were described in earlier chapters:

- ▶ ICAL Synchronous Program Switch COBOL program
- ▶ Asynchronous callout to a Stateless Session bean
- ▶ Asynchronous callout to a Message Driven bean
- ▶ Synchronous callout to a Stateless Session bean
- ▶ Synchronous callout to a Message Driven bean
- ▶ Feed from an IMS application in MashupHub
- ▶ WSDL files for a DLIModel generated web service
- ▶ XSD files for a DLIModel generated web service
- ▶ Enhanced Provider MPP template sample

A.1 ICAL Synchronous Program Switch COBOL program

Example A.1 contains the source code for the COBOL application that is used in “Scenario 3: Rejection of input after GFL is reached” on page 141.

Example A-1 Source and assembly plus bind jobs for COBOL ICAL Synchronous Program Switch program IAPMDI27

```
//IAPMDI27 JOB 248001,CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1),
//          REGION=4M
/*ROUTE PRINT THISCPU/IMSTST75
//*****
//COBOL1 EXEC PGM=IGYCRCTL,
//          PARM='TEST,NODYNAM,LIB,MAP,OBJECT,RES,APOST,XREF'
//STEPLIB DD DSN=IGYV4R20.SIGYCOMP,DISP=SHR
//SYSLIB DD DSN=CEE.SCEERUN,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSLIN DD DSN=&&LOADSET,DISP=(MOD,PASS),
//          UNIT=SYSDA,SPACE=(80,(1500,600))
//SYSUT1 DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT2 DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT3 DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT4 DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT5 DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT6 DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSUT7 DD UNIT=SYSDA,SPACE=(460,(350,100))
//SYSIN DD *
IDENTIFICATION DIVISION.
PROGRAM-ID.          IAPMDI27
*          COBOL370
*REMARKS.
* Issues ICAL Synchronous Program Switch
* Input Message: APOL11
* Switch to      :
* Notes          : This application expects the OTMA descriptor
*                  passed as a parameter
*
ENVIRONMENT DIVISION.
DATA DIVISION.

WORKING-STORAGE SECTION.
77 FILLER          PIC X(16) VALUE '*** BEGIN WS ***'.

*****
*          IMS DECLARATIONS
*****

77 QC              PIC X(2) VALUE 'QC'.
77 GU-FUNC         PIC X(4) VALUE 'GU '.
77 ICAL            PIC X(4) VALUE 'ICAL'.
77 ISRT           PIC X(4) VALUE 'ISRT'.
77 ROLL-FUNC       PIC X(4) VALUE 'ROLL'.
77 DISPLAY-LL      PIC 99999.
77 DISPLAY-ZZ      PIC 99999.
77 ERROR-STATUS    PIC X(2).
77 AIBRSFLD-EDITED PIC ZZZZZZ9.
```



```

77 AIBOALEN-EDITED      PIC ZZZZZZ9.
77 AIBOAUSE-EDITED      PIC ZZZZZZ9.
77 ICALSUBFUNC          PIC x(8) VALUE 'SENDREC'.
77 ATIMEFLD             PIC X(4) VALUE '1200'.
77 SCARSNM1             PIC X(8) VALUE 'OTMDEST1'.

```

01 SWITCHES.

```

05 MESSAGE-FOUND-SW     PIC X VALUE 'N'.
08 MESSAGE-FOUND        VALUE 'Y'.
05 END-OF-MESSAGES-SW   PIC X VALUE 'N'.
08 END-OF-MESSAGES      VALUE 'Y'.

```

01 IN-MESSAGE.

```

03 IN-LL               PIC S9(3) COMP.
03 IN-ZZ               PIC S9(3) COMP.
03 IN-TRANCODE         PIC X(9).
03 IN-OTMADESC         PIC X(9).
03 SW-TRANCODE         PIC X(9).
03 MY-DATA1            PIC X(9).
03 MY-DATA2            PIC X(9).

```

* AIB

01 AIB.

```

02 AIBRID PIC x(8) VALUE 'DFS AIB '.
02 AIBRLN PIC 9(9) USAGE BINARY.
02 AIBSFUNC PIC x(8).
02 AIBRSNM1 PIC x(8).
02 AIBRSNM2 PIC x(8).
02 AIBRESV1 PIC x(8).
02 AIBOALEN PIC 9(9) USAGE BINARY.
02 AIBOAUSE PIC 9(9) USAGE BINARY.
02 AIBRSFLD PIC 9(9) USAGE BINARY.
02 AIBRESV2 PIC x(8).
02 AIBRETRN PIC 9(9) USAGE BINARY.
02 AIBREASN PIC 9(9) USAGE BINARY.
02 AIBERRXT PIC 9(9) USAGE BINARY.
02 AIBRESA1 USAGE POINTER.
02 AIBRESA2 USAGE POINTER.
02 AIBRESA3 USAGE POINTER.
02 AIBRESV4 PIC x(40).
02 AIBRSAVE OCCURS 18 TIMES USAGE POINTER.
02 AIBRTOKN OCCURS 6 TIMES USAGE POINTER.
02 AIBRTOKC PIC x(16).
02 AIBRTOKV PIC x(16).
02 AIBRTOKA OCCURS 2 TIMES PIC 9(9) USAGE BINARY.

```

01 FORMATEXT.

```

02 TEXT3              PIC X(50).

```

01 DC-TEXT0.

```

02 TEXT2              PIC X(14) VALUE 'DLI CALL : '.
02 ERROR-CALL         PIC X(4) VALUE SPACES.

```

01 DC-TEXT1.

```

02 TEXT1          PIC X(14) VALUE 'RETURN CODE : '.
02 ERR-AIBRETRN   PIC ZZZ9.

01 DC-TEXT2.
02 TEXT1          PIC X(14) VALUE 'REASON CODE : '.
02 ERR-AIBREASN   PIC ZZZ9.

01 DC-TEXT3.
02 TEXT1          PIC X(14) VALUE 'ERREXT CODE : '.
02 ERR-AIBERRXT   PIC ZZZ9.

* DATA AREA OUTPUT

01 OUTPUT-AREA.
02 OUT-LL         PICTURE S9(3) COMP VALUE +70.
02 OUT-ZZ         PICTURE S9(3) COMP VALUE +0.
02 OUT-LINE       PICTURE X(70) VALUE SPACES.
02 OUT-DATA REDEFINES OUT-LINE.
04 OUT-MSG1       PIC X(16).
04 OUT-MSG2       PIC X(10).
04 OUT-MSG3       PIC X(10).
04 OUT-MSG4       PIC X(34).

* ERROR DATA AREA OUTPUT

01 ERROR-OUTPUT-AREA.
02 ERR-LL         PICTURE S9(3) COMP VALUE +62.
02 ERR-ZZ         PICTURE S9(3) COMP VALUE +0.
02 ERR-TEXT1 PIC X(14) VALUE 'RETURN CODE : '.
02 ERR-OUTRETRN   PIC ZZZ9.
02 ERR-TEXT2 PIC X(15) VALUE 'REASON CODE : '.
02 ERR-OUTREASN   PIC ZZZ9.
02 ERR-TEXT3 PIC X(15) VALUE 'ERREXT CODE : '.
02 ERR-OUTERRXT   PIC ZZZ9.

* ERROR DATA AREA OUTPUT1

01 ERROR-OUTPUT-TEXT.
02 ERT-LL         PICTURE S9(3) COMP VALUE +60.
02 ERT-ZZ         PICTURE S9(3) COMP VALUE +0.
02 ERT-TEXT1 PIC X(54) VALUE SPACES.

* ICAL Request Area
01 CALL-REQUEST.
02 CALL-LL        PICTURE S9(3) COMP VALUE +24.
02 CALL-ZZ        PICTURE S9(3) COMP VALUE +0.
02 CALL-TRAN      PIC X(8) VALUE SPACES.
02 CALL-DATA      PIC X(12) VALUE SPACES.

* ICAL Response Area
01 CALL-RESP.
02 RESP-LL        PICTURE S9(3) COMP VALUE +24.
02 RESP-ZZ        PICTURE S9(3) COMP VALUE +0.
02 CA-RESPONSE    PIC X(20) VALUE SPACES.

```

```

LINKAGE SECTION.
01 IOPCB.
    02 IO-LTERM          PIC X(8).
    02 IO-RESV           PIC X(2).
    02 IO-STATUS         PIC X(2).
    02 IO-PREF           PIC X(12).
    02 IO-MODN           PIC X(8).
    02 IO-USERID         PIC X(8).
    02 IO-GROUPID        PIC X(8).

PROCEDURE DIVISION.
*
* Main Program
    ENTRY 'DLITCBL' USING IOPCB.
    DISPLAY "IAPMDI27 Execution begins... ".
    PERFORM PROCESS-INPUT-MESSAGE
    IF IO-STATUS = SPACES
        PERFORM ICAL-SWITCH.
    PERFORM INSERT-IO

    GOBACK.
*
*
* PROCEDURE PROCESS-INPUT-MESSAGE
PROCESS-INPUT-MESSAGE.
*
    MOVE "$$" TO IO-STATUS.
    CALL 'CBLTDLI' USING GU-FUNC, IOPCB, IN-MESSAGE.
    IF IO-STATUS = SPACES
        DISPLAY "TRANCODE :" IN-TRANCODE
        DISPLAY "OTMADESC :" IN-OTMADESC
        DISPLAY "SWI TRAN :" SW-TRANCODE
        DISPLAY "MY-DATA1 :" MY-DATA1
        MOVE 'Y' TO MESSAGE-FOUND-SW
        MOVE 'N' TO END-OF-MESSAGES-SW
    ELSE
        MOVE GU-FUNC TO ERROR-CALL
        MOVE 'N' TO MESSAGE-FOUND-SW
        MOVE 'Y' TO END-OF-MESSAGES-SW.
*
* PROCEDURE ICAL-SWITCH
* Set up to make ICAL call
* Set AIBLEN, AIB Sub Function(SENDRECV), OTMA Descriptor to use,
* Transaction to switch to, Data associated with transaction
* Wait Time to AIBRSFLD, Length of request and response areas,
ICAL-SWITCH.
    MOVE LENGTH OF AIB TO AIBRLen.
    MOVE ICALSUBFUNC TO AIBSFUNC.
    MOVE IN-OTMADESC(1:8) TO AIBRSNM1.
    MOVE SW-TRANCODE(1:8) TO CALL-TRAN.
    MOVE MY-DATA1 TO CALL-DATA
    MOVE 2500 TO AIBRSFLD.
    MOVE LENGTH OF CALL-REQUEST TO AIBOALEN.

```

```

MOVE LENGTH OF CALL-RESP TO AIBOAUSE.
MOVE AIBOALEN TO AIBOALEN-EDITED.
MOVE AIBOAUSE TO AIBOAUSE-EDITED.
MOVE AIBRSFLD TO AIBRSFLD-EDITED.
DISPLAY 'ICAL REQUEST LEN : ' AIBOALEN-EDITED UPON CONSOLE.
DISPLAY 'ICAL RESPONSE LEN : ' AIBOAUSE-EDITED UPON CONSOLE.
DISPLAY "Calling ICAL with : " CALL-TRAN " " CALL-DATA.
CALL 'AIBTDLI' USING ICAL, AIB, CALL-REQUEST, CALL-RESP.
IF AIBRETRN NOT EQUAL ZEROES
MOVE IO-STATUS TO ERROR-STATUS.
MOVE ICAL TO ERROR-CALL.
MOVE AIBOALEN TO AIBOALEN-EDITED.
MOVE AIBOAUSE TO AIBOAUSE-EDITED.
MOVE AIBRETRN TO ERR-AIBRETRN.
MOVE AIBREASN TO ERR-AIBREASN.
MOVE AIBERRXT TO ERR-AIBERRXT.
DISPLAY '+++++' UPON CONSOLE.
DISPLAY 'PGM SWITCH TO TRAN : ' CALL-TRAN UPON CONSOLE.
DISPLAY 'ICAL SUB-FUNCTION : ' AIBSFUNC UPON CONSOLE.
DISPLAY 'OTMA DESCRIPTOR : ' AIBRSNM1 UPON CONSOLE.
DISPLAY 'ICAL TIMEOUT VALUE : ' AIBRSFLD-EDITED UPON CONSOLE.
DISPLAY 'ACTUAL REQUEST LEN : ' AIBOALEN-EDITED UPON CONSOLE.
DISPLAY 'ACTUAL RESPONSE LEN : ' AIBOAUSE-EDITED UPON CONSOLE.
DISPLAY '+++++' UPON CONSOLE.
DISPLAY 'RESP MESSAGE : ' CA-RESPONSE UPON CONSOLE.
DISPLAY '+++++' UPON CONSOLE.
PERFORM WRITE-DC-TEXT.
DISPLAY '+++++' UPON CONSOLE.
DISPLAY '+++++' UPON CONSOLE.
* END-IF.

```

* PROCEDURE WRITE-DC-TEXT : WRITE ERROR STATUS CODE

```

WRITE-DC-TEXT.
DISPLAY "In WRITE-DC ...".
MOVE IO-STATUS TO ERROR-STATUS.
DISPLAY DC-TEXT0 UPON CONSOLE.
DISPLAY DC-TEXT1 UPON CONSOLE.
DISPLAY DC-TEXT2 UPON CONSOLE.
DISPLAY DC-TEXT3 UPON CONSOLE.

```

```

WRITE-FORMAT-TEXT.
DISPLAY "In WRITE-INFO ...".
DISPLAY DC-TEXT0 UPON CONSOLE.
DISPLAY FORMATEX UPON CONSOLE.

```

* PROCEDURE INSERT-IO : INSERT FOR IOPCB REQUEST HANDLER

```

INSERT-IO.
DISPLAY "Calling ISRT ...".
MOVE 'MESSAGE SWITCH' TO OUT-MSG1.
IF IO-STATUS = SPACES
MOVE CA-RESPONSE(1:AIBOAUSE) TO OUT-LINE
MOVE AIBOAUSE TO OUT-LL
ELSE

```

```

        MOVE 'FAILED' TO OUT-MSG2.
        CALL 'CBLTDLI' USING ISRT, IOPCB, OUTPUT-AREA.
        IF IO-STATUS NOT = SPACES
            THEN PERFORM WRITE-DC-TEXT.

/*
/*
//LKED      EXEC PGM=IEWL,COND=(8,LE,COBOL1),
//          PARM='LIST,XREF,LET,RENT,MAP'
//SYSLIB    DD DSN=IMSB LD.I%%TS%%.CRESLIB,DISP=SHR
//          DD DSN=SYS1.MACLIB,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//          DD DSN=CEE.SCEELKED,DISP=SHR
//          DD DSN=IMSTESTL.TNUCT,DISP=SHR
//RESLIB    DD DSN=IMSB LD.I%%TS%%.CRESLIB,DISP=SHR
//SYSLMOD   DD DSN=IMSTESTL.TNUCT,DISP=SHR
//SYSUT1    DD UNIT=SYSDA,DCB=BLKSIZE=1024,SPACE=(1024,(200,20))
//SYSPRINT  DD SYSOUT=*
//SYSLIN    DD DSN=&&LOADSET,DISP=(OLD,DELETE)
//          DD DDNAME=SYSIN
//SYSIN     DD *
            ENTRY DLITCBL
            NAME IAPMDI27(R)
/*

```

A.2 Asynchronous callout to a Stateless Session bean

Example A-2 is referenced in “Asynchronous callout to an SLSB” on page 242. The code is offered *as is*. Some details can be changed after the availability date.

Example A-2 Code snippet for asynchronous to SLSB in WebSphere Application Server

```

package ejbs;
import javax.naming.InitialContext;
import javax.resource.cci.Connection;
import javax.resource.cci.ConnectionFactory;
import javax.resource.cci.Interaction;
import javax.resource.cci.Record;
import com.ibm.connector2.ims.ico.IMSConnectionSpec;
import com.ibm.connector2.ims.ico.IMSInteractionSpec;
/**
 * bean implementation class for Enterprise bean: SLSB4IMSASyncCallOut1
 */
public class SLSB4IMSASyncCallOut1Bean implements javax.ejb.SessionBean {
    /**
     * deleted lines without interest for IMS callout
     */
    static final long serialVersionUID = 3206093459760846163L;
    public void start() { //_____ startup EJB called method
        try {
            InitialContext initCtx = new InitialContext(); // (1)
            // JNDI lookup returns connFactory
            ConnectionFactory connFactory = (ConnectionFactory) initCtx
                .lookup("java:comp/env/ibm/ims/IMSTarget"); // -> ConnectionFactory (2)

```

```

IMSConnectionSpec connSpec = new IMSConnectionSpec(); // (3)
Connection connection = connFactory.getConnection(connSpec); // (4)
Interaction interaction = connection.createInteraction(); // (5)
IMSInteractionSpec interactionSpec = new IMSInteractionSpec(); // (6)
// set the commit mode and sync level
interactionSpec.setCommitMode(0); // (7)
interactionSpec.setSyncLevel(1); // (7)
// An 8-character queue name that the asynchronous messages to be retrieved from
String calloutQueueName = new String("CALLOUTQ");
// Set the asynchronous queue name for the callout message
interactionSpec.setAltClientID(calloutQueueName); // (8)
// Set interactionVerb to retrieve async output with a timeout
interactionSpec
    .setInteractionVerb(IMSInteractionSpec.SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT); //(9)
interactionSpec.setExecutionTimeout(3600000);
Record calloutMsg = null; // (9) should be a DataBean
// looping for message from IMS
for (;;) {
    try {
        // Execute the interaction
        interaction.execute(interactionSpec, null, calloutMsg); // (10)
        // process received message
        // .....
        // ----- if response required -----
        // Use again a CCI interaction for sending back the response: SEND_ONLY
        // Response should contain IMSTRAN, it is a new transaction
        //-----
        // build new interaction if required to send response IMSTRAN // (11)
        //-----}
    } catch (Exception e2) {
        // if the exception is an execution timeout error,
        // you can either do nothing and continue to loop
        // or process the error and then break the loop
        break;
    }
}
interaction.close();
connection.close();
} catch (Exception e1) {
}
}
}

```

The number in the parentheses in Example A-2 on page 435 refer to the following actions:

1. Set the initial context.
2. Use the context to look up the Connection Factory with JNDIName "java:comp/env/ibm/ims/IMSTarget".
3. Instantiate connectionSpec.
4. Get the connection from connectionFactory, which is augmented with connectionSpec.
5. Create an interaction from the connection.
6. Instantiate interactionSpec.
7. Set the synclevel and commit as indicated in interactionSpec.

8. Set the destination for reading callout message in interactionSpec.
9. Set interactionVerb.
10. The message must be received in a databean (javax.resource.cci.Record).
11. Run the interaction with IMS to retrieve the callout message.
12. If required, build a response message using the existing connection to send a new transaction to IMS as a "send_only" interaction.

A.3 Asynchronous callout to a Message Driven bean

Example A-3 is referenced at "Asynchronous callout to an MDB" on page 243. The code is offered *as is*. Some details can be changed after the availability date.

Example: A-3 Code snippet for asynchronous callout to MDB in WebSphere Application Server

```
package ejbs;
import javax.resource.cci.Record;
/**
 * Bean implementation class for Enterprise bean: MDB4IMSASyncCallOut1
 */
public class MDB4IMSASyncCallOut1Bean
    implements
        javax.ejb.MessageDrivenBean,
        commonj.connector.runtime.InboundListener { // (1)
    private javax.ejb.MessageDrivenContext fMessageDrivenCtx;
    /**
     * deleted lines without interest for IMS callout
     */
    /**
     * onMessage NOT used for asynchronous call
     */
    public javax.resource.cci.Record onMessage(javax.resource.cci.Record arg0){
        return null;
    }
    /**
     * onNotification
     */
    public void onNotification(javax.resource.cci.Record arg0)
        throws javax.resource.ResourceException { // (2)
        processMessageReceived(arg0); // (3)
        // ----- if response required -----
        // Use again a CCI interaction for sending back the response: SEND_ONLY
        // Response should contain IMSTRAN, it is a new transaction (4)
        //-----
        // build new interaction if required to send response IMSTRAN
        //-----}
    }
    /**
     * processMessageReceived
     */
    public void processMessageReceived(Object event){
        // examine the message and process accordingly
    }
}
```

```
}  
}
```

The numbers in the parentheses in Example A-3 on page 437 refer to the following actions:

1. The MDB must implement this interface.
2. The message is the type (`javax.resource.cci.Record`) received in `onNotification()`.
3. Process the received message.
4. If required, build a response message and set up a connection to send a new transaction to IMS as a “send_only” interaction.

The MDB code is more simple than the SLSB code because a listener is used. This listener must know where to listen. This information is declared in the *ActivationSpec*, as shown in Figure A-1. The *ActivationSpec* is one of the elements of the MDB.

J2C activation specifications

Use this page to configure a J2C activation specification, which is used by the resource adapter when configuring a specific endpoint instance that configures one or more endpoints must specify the resource adapter that sends messages to the endpoint, and it must use the activation specification to provide the configuration properties for processing the inbound message.

☒ Scope: Cell=cl6671, Node=nd6671

Scope specifies the level at which the resource definition is visible. For detailed information on what scope is and how it works, [see the scope settings help](#)

Node=nd6671

Select	Name	JNDI name	Scope	Provider	Description	Message listener type
<input type="checkbox"/>	MDB4IMSASyncCallOut1ActSpec	eis/MDB4IMSASyncCallOut1	Node=nd6671	IMS TM Resource Adapter	ActivationSpec for MDB EJB2.1	com.ibm.j2ca.base.ExtendedMessageListener

Figure A-1 *ActivationSpec*

For each MDB, an activationSpec must be defined by the administrator of the WebSphere Application Server. It is declared under the TM Resource Adapter provider, as shown in Figure A-2.

J2C activation specifications

J2C activation specifications > MDB4IMSAyncCallOut1ActSpec > Custom properties

Use this page to specify custom properties that your enterprise information system (EIS) requires for the resource providers and resource factories that you configure. For example, most database vendors require additional custom properties for data sources that access the database.

⊞ Preferences

Name	Value	Description	Required
queueNames			true
portNumber	9999		true
hostName	wtsc67.itso.ibm.com		true
dataStoreName	IMSL		true
SSLKeyStoreName			false
retryInterval	60000		false
SSLKeyStorePassword			false
filterFutureEvents	false		false
pollPeriod	2000		false
pollQuantity	10		false
BONamespace			false
deliveryType	ORDERED		false
SSLEnabled	false		false
assuredOnceDelivery	true		false
stopPollingOnError	false		false
eventTypeFilter			false
userName			false
retryLimit	0		false
SSLTrustStorePassword			false
SSLTrustStoreName			false

Page: 1 of 2
Total 25

Figure A-2 ActivationSpec properties

In Figure A-3, you find the properties that must be specified for the activationSpec of the MDB. Mainly, they correspond to the IP address, the port number of IMS Connect, and the IMS data store name.

WebSphere Bindings
The following are binding properties for the WebSphere Application Server.

☐ Listener Port

Listener port name:

☒ JCA Adapter

ActivationSpec JNDI name:

ActivationSpec Authorization Alias:

Destination JNDI name:

Figure A-3 ActivationSpec name in Deployment Descriptor of MDB

The link between activationSpec and MDB is through the EJB (MDB) deployment descriptor, which is part of the deployed Java EE application together with the MDB itself.

A.4 Synchronous callout to a Stateless Session bean

Example A-4 is referenced in “Synchronous callout to an SLSB” on page 247. The code is offered *as is*. Some details can be changed after the availability date.

Example: A-4 Code snippet for synchronous callout to SLDB in WebSphere Application Server

```
package ejbs;
import javax.naming.InitialContext;
import javax.resource.cci.Connection;
import javax.resource.cci.ConnectionFactory;
import javax.resource.cci.Interaction;
import javax.resource.cci.Record;
import com.ibm.connector2.ims.ico.IMSConnectionSpec;
import com.ibm.connector2.ims.ico.IMSInteractionSpec;
/**
 * Bean implementation class for Enterprise bean: SLSB4IMSSyncCallOut1
 */
public class SLSB4IMSSyncCallOut1Bean implements javax.ejb.SessionBean {

    static final long serialVersionUID = 3206093459760846163L;
    private javax.ejb.SessionContext mySessionCtx;
    /**
     * deleted lines without interest for IMS callout
     */
    public void start() {
        try {
            InitialContext initCtx = new InitialContext(); // (1)
            // JNDI lookup returns connFactory
            ConnectionFactory connFactory = (ConnectionFactory) initCtx
                .lookup("java:comp/env/ibm/ims/IMSTarget"); // (2)
            IMSConnectionSpec connSpec = new IMSConnectionSpec(); // (3)
```

```

Connection connection = connFactory.getConnection(connSpec); // (4)
Interaction interaction = connection.createInteraction(); // (5)
IMSInteractionSpec interactionSpec = new IMSInteractionSpec(); // (6)
// set the commit mode and sync level
interactionSpec.setCommitMode(0); // (7)
interactionSpec.setSyncLevel(1); // (7)
// A 8 character name that the asynchronous messages to be retrieved from
String calloutQueueName = new String("CALLOUTQ");
// Set the queue name for the callout message
interactionSpec.setAltClientID(calloutQueueName); // (8)
// Set InteractionVerb for retrieve async output with a very large // timeout
interactionSpec
    .setInteractionVerb(IMSInteractionSpec.SYNC_RECEIVE_ASYNCOUTPUT_SINGLE_WAIT); // (9)
interactionSpec.setExecutionTimeout(360000);
// Execute the interaction
Record calloutMsg = null; /* Temporary should be databean */ // (10)
// looping for message from IMS
for (;;) {
    try {
        interaction.execute(interactionSpec, null, calloutMsg); // (11)
        // Get Correlation token
        // ----> following is for synchronous callout
        String corrToken = interactionSpec.getSyncCalloutToken(); // (12)
        // Further processing on the calloutMsg
        // Send back the response
        interactionSpec.setInteractionVerb
            (com.ibm.connector2.ims.ico.IMSInteractionSpec.SYNC_SEND); // (13)
        // ----> following is correlating callout and response, if new InteractionSpec
        interactionSpec.setSyncCalloutToken(corrToken); // (14)
        // Execute the interaction
        Record responseMsg = null; /* Temporary */
        interaction.execute(interactionSpec, responseMsg, null); // (15)
        interaction.close();
        connection.close();
    } catch (Exception e1) {
    }
}
}
}

```

The numbers in the parentheses in Example A-4 on page 440 refer to the following actions:

1. Set the initial context.
2. Use the context to look up the Connection Factory with JNDIName "java:comp/env/ibm/ims/IMSTarget".
3. Instantiate connectionSpec.
4. Get the connection from connectionFactory.
5. Create an interaction from the connection.
6. Instantiate interactionSpec.
7. Set the synclevel and commit as indicated in interactionSpec.
8. Set the destination for reading the callout message in interactionSpec.
9. Set interactionVerb.
10. The message must be received in a databean (javax.resource.cci.Record).

11. Run an interaction with IMS to retrieve the callout message.
12. Recuperate the correlation token.
13. Set interactionVerb for the sending response.
14. Set the correlation token.
15. Build a response message of type databean (javax.resource.cci.Record), and use the existing connection to send a new transaction to IMS as a “send_only” interaction.

A.5 Synchronous callout to a Message Driven bean

Example A-5 is referenced at “Synchronous callout to an MDB” on page 248. The code is offered *as is*. Some details can be changed after the availability date.

Example: A-5 Code snippet for synchronous callout to MDB in WebSphere Application Server

```
package ejbs;
import javax.resource.cci.Record;
/**
 * Bean implementation class for Enterprise bean: MDB4IMSSyncCallOut1
 */
public class MDB4IMSSyncCallOut1Bean
    implements
        javax.ejb.MessageDrivenBean,
        commonj.connector.runtime.InboundListener { // (1)
    private static final long serialVersionUID = 1L;
    private javax.ejb.MessageDrivenContext fMessageDrivenCtx;
    /**
     * deleted lines without interest for IMS callout
     */
    /**
     * onMessage
     */
    public javax.resource.cci.Record onMessage(javax.resource.cci.Record arg0){ // (2)
        processMessageReceived(arg0); // (3)
        Record testRecord = null; /* temporarily */ // (4)
        //INPUTMSG testRecord = new INPUTMSG();
        //testRecord.setIn__ll((short)74);
        //testRecord.setIn__zz((short)0);
        //testRecord.setIn__data(...);
        return testRecord; // (5)
    }
    /**
     * onNotification NOT used for synchronous call
     */
    public void onNotification(javax.resource.cci.Record arg0)
        throws javax.resource.ResourceException {
    }
    /**
     * processMessageReceived
     */
    public void processMessageReceived(Object event){
        // examine the message and process accordingly
    }
}
```

}

The numbers in the parentheses in Example A-5 on page 442 refer to the following actions:

1. The MDB must implement this interface.
2. A message of type `javax.resource.cci.Record` is received in `onMessage()`.
3. Process the received message.
4. Prepare a response record of type `databean (javax.resource.cci.Record)`.
5. Return a response (automatically correlated with a request).

The MDB code is more simple than the SLSB code because a listener is used. This listener must know where to listen. This information is declared in the *ActivationSpec*. The *ActivationSpec* is one of the accompanying elements of the MDB. For each MDB, an *ActivationSpec* must be defined by the administration of WebSphere Application Server. It is declared under the TM Resource Adapter provider.

A.6 Feed from an IMS application in MashupHub

Example A-6 is referenced in 6.1.1, “Developing Java applications” on page 216. The code is offered *as is*.

Example A-6 Example of an IMS feed in MashupHub

```
<?xml version="1.0" encoding="UTF-8"?>
<feed xmlns="http://www.w3.org/2005/Atom">
  <id>https://server.company.com:9443/mashuphub/client/plugin/generate/
entryid/304/pluginid/7</id>
  <subtitle type="text">A list of insurance policy holders</subtitle>
  <link href="https://server.company.com:9443/mashuphub/client/plugin/generate/
entryid/304/pluginid/7" rel="self" type="application/atom+xml"></link>
  <updated>2008-05-28T21:10:14.189Z</updated>
  <title type="text">Insurance Policy Holders</title>
  <generator>InfoSphere MashupHub</generator>

  <entry xmlns="http://www.w3.org/2005/Atom">
    <title type="text">Item 1</title>
    <id>urn:uuid:1</id>
    <updated>2008-05-28T21:10:14.189Z</updated>
    <author>
      <name>authorname@company.com</name>
    </author>
    <summary type="text">Atom Feed entry 1</summary>
    <content type="application/xml">
      <RepeatingElement
xmlns="http://www.ibm.com/xmlns/atom/content/imsrecord/1.0">
        <POLICYDATA>
          <out_ll>964</out_ll>
          <out_zz>512</out_zz>
          <policy_detail>
            <policyid>AM4470</policyid>
            <customername>Alva Morua</customername>
            <coverage>104021.00</coverage>
            <premium>184.00</premium>
```

```

        <startdate>2006-10-23</startdate>
        <enddate>2007-05-23</enddate>
        <description>For physical loss or damage, coverage includes the hull,
machinery, fittings, furnishings, and permanently attached equipment for an agreed
value. These policies also provide broader liability protection than a
homeownerspolicy.</description>
        <paddress>11151 pine st</paddress>
        <city>Dixie</city>
        <state>Florida</state>
    </policy_detail>
    <policy_detail>
        <policyid>CG7010</policyid>
        <customername>Cruz Gorell</customername>
        <coverage>451171.00</coverage>
        <premium>775.00</premium>
        <startdate>2006-09-18</startdate>
        <enddate>2006-12-18</enddate>
        <description>The policy coverage provides compensation liability for
injury to persons employed by you who may work on your boat, but are not crew
members. Mechanics, carpenters, painters, and cleaners are included in this
category.</description>
        <paddress>11285 riverside dr</paddress>
        <city>Arkansas</city>
        <state>Arkansas</state>
    </policy_detail>
    <policy_detail>
        <policyid>CS5000</policyid>
        <customername>Clark Sandigo</customername>
        <coverage>363580.00</coverage>
        <premium>1395.00</premium>
        <startdate>2006-09-20</startdate>
        <enddate>2008-04-20</enddate>
        <description>The policy covers physical damage to the hull, sails,
machinery, furniture, and most other equipment that is normally used on board.
Most perials results from latent defects of workm</description>
        <paddress>63979 tree blvd</paddress>
        <city>Presidio Valley</city>
        <state>Texas</state>
    </policy_detail>
    <policy_detail>
        <policyid>CS8416</policyid>
        <customername>Chase Saucedo</customername>
        <coverage>424210.99</coverage>
        <premium>289.00</premium>
        <startdate>2006-10-19</startdate>
        <enddate>2008-03-19</enddate>
        <description>The policy coverage provides compensation liability for
injury to persons employed by you who may work on your boat, but are not crew
members. Mechanics, carpenters, painters, and cleaners are included in this
category.</description>
        <paddress>82814 oak st</paddress>
        <city>Lafayette</city>
        <state>Arkansas</state>
    </policy_detail>
    <policy_detail>

```

```

        <policyid>FC2267</policyid>
        <customername>Foster Cadaviec</customername>
        <coverage>310821.00</coverage>
        <premium>219.00</premium>
        <startdate>2006-10-27</startdate>
        <enddate>2007-04027</enddate>
        <description>For physical loss or damage, coverage includes the hull,
machinery, fittings, furnishings and permanently attached equipment for an agreed
value. These policies also provide broader liability protection than a homeowners
policy.</description>
        <paddress>38356 tree blvd</paddress>
        <city>Dawson</city>
        <state>Florida</state>
    </policy_detail>
</POLICYDATA>
</RepeatingElement>
</content>
</entry>
</feed>

```

A.7 WSDL files for a DLIModel generated web service

Example A-7 is referenced in 6.1, “Java Platform, Enterprise Edition” on page 216. The code is offered *as is*. Example A-7 shows a DLIModel-generated WSDL.

Example: A-7 DealersModels.wsdl

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:ims="http://www.ibm.com/ims/das/config" xmlns="http://ibm.sj.dealership"
targetNamespace="http://ibm.sj.dealership">
<!--Types-->
<wsdl:types>
<xsd:schema targetNamespace="http://ibm.sj.dealership">
<import xmlns="http://www.w3.org/2001/XMLSchema" schemaLocation="AUTPSB11-AUTS2PCB.xsd"
namespace="http://www.ibm.com/ims/AUTPSB11/AUTS2PCB"/> // <----- pointer to Schema Location see Example
16-4
<xsd:element name="getDealerModels">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="DLRNO">
<xsd:simpleType>
<xsd:restriction base="xsd:string">
<xsd:length xmlns="http://www.ibm.com/ims/das/config" value="4"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="getDealerModelsResponse">
<xsd:complexType>
<xsd:sequence>
<element xmlns="http://www.w3.org/2001/XMLSchema" xmlns:pcb="http://www.ibm.com/ims/AUTPSB11/AUTS2PCB"
ref="pcb:MODEL" minOccurs="0" maxOccurs="1"/>

```

```

</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</wsdl:types>
<!--Messages-->
<wsdl:message name="getDealerModelsRequest">
<wsdl:part name="parameters" element="getDealerModels"/>
</wsdl:message>
<wsdl:message name="getDealerModelsResponse">
<wsdl:part name="parameters" element="getDealerModelsResponse"/>
</wsdl:message>
<!--Ports-->
<wsdl:portType name="DealerModels">
<wsdl:operation name="getDealerModels">
<wsdl:input name="getDealerModelsRequest" message="getDealerModelsRequest"/>
<wsdl:output name="getDealerModelsResponse" message="getDealerModelsResponse"/>
</wsdl:operation>
</wsdl:portType>
<!--Bindings-->
<wsdl:binding type="DealerModels" name="DealerModelsSoapBinding">
<wsaw:UsingAddressing xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" wsdl:required="false"/>
<wsdlsoap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<wsdl:operation name="getDealerModels">
<wsdlsoap:operation soapAction="getDealerModels"/>
<wsdl:input name="getDealerModelsRequest">
<wsdlsoap:body use="literal"/>
</wsdl:input>
<wsdl:output name="getDealerModelsResponse">
<wsdlsoap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<!--Services-->
<wsdl:service name="DealerModels_Services">
<wsdl:port name="DealerModels" binding="DealerModelsSoapBinding">
<wsdlsoap:address location="http://localhost:9080/DealerModels/services/DealerModels"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

A.8 XSD files for a DLIModel generated web service

Example A-8 is referenced in 6.1, “Java Platform, Enterprise Edition” on page 216.
 Example A-7 on page 445 references the XSD files in Example A-8.

Example: A-8 AUTPSB11-AUTS2PCB.xsd

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:ims="http://www.ibm.com/ims"
  xmlns="http://www.ibm.com/ims/AUTPSB11/AUTS2PCB"
  targetNamespace="http://www.ibm.com/ims/AUTPSB11/AUTS2PCB"
  elementFormDefault="qualified">

  <xsd:annotation>
    <xsd:appinfo>

```



```

        <ims:DLI version="2.0" mode="retrieve" topLevelElement="AUTS2PCB" />
    </xsd:appinfo>
</xsd:annotation>

<!-- AUTS2PCB -->
<xsd:element name="AUTS2PCB">
    <xsd:annotation>
        <xsd:appinfo>
            <ims:pcb name="AUTS2PCB" alias="AUTS2PCB"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element ref="DEALER" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

<!-- DEALER -->
<xsd:element name="DEALER">
    <xsd:annotation>
        <xsd:appinfo>
            <ims:segment name="DEALER" alias="DEALER"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="DLRNO" >
                <xsd:simpleType>
                    <xsd:annotation>
                        <xsd:appinfo>
                            <ims:field name="DLRNO" alias="DLRNO"/>
                        </xsd:appinfo>
                    </xsd:annotation>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="4"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="DLRNAME" >
                <xsd:simpleType>
                    <xsd:annotation>
                        <xsd:appinfo>
                            <ims:field name="DLRNAME" alias="DLRNAME"/>
                        </xsd:appinfo>
                    </xsd:annotation>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="30"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="CITY" >
                <xsd:simpleType>
                    <xsd:annotation>
                        <xsd:appinfo>

```

```

        <ims:field name="CITY" alias="CITY"/>
    </xsd:appinfo>
</xsd:annotation>
<xsd:restriction base="xsd:string">
    <xsd:maxLength value="10"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="ZIP" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="ZIP" alias="ZIP"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="PHONE" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="PHONE" alias="PHONE"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="7"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element ref="MODEL" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<!-- MODEL -->
<xsd:element name="MODEL">
    <xsd:annotation>
        <xsd:appinfo>
            <ims:segment name="MODEL" alias="MODEL"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="MODKEY" >
                <xsd:simpleType>
                    <xsd:annotation>
                        <xsd:appinfo>
                            <ims:field name="MODKEY" alias="MODKEY"/>
                        </xsd:appinfo>
                    </xsd:annotation>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="24"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

```

```

        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="MODTYPE" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="MODTYPE" alias="MODTYPE"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="2"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="MAKE" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="MAKE" alias="MAKE"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="MODEL" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="MODEL" alias="MODEL"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="YEAR" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="YEAR" alias="YEAR"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="4"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="MSRP" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>

```

```

        <ims:field name="MSRP" alias="MSRP"/>
    </xsd:appinfo>
</xsd:annotation>
    <xsd:restriction base="xsd:string">
        <xsd:maxLength value="5"/>
    </xsd:restriction>
</xsd:simpleType>
</xsd:element>
<xsd:element name="COUNT1" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="COUNT" alias="COUNT1"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="2"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
    <xsd:element ref="STOCK" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>

<!-- STOCK -->
<xsd:element name="STOCK">
    <xsd:annotation>
        <xsd:appinfo>
            <ims:segment name="STOCK" alias="STOCK"/>
        </xsd:appinfo>
    </xsd:annotation>
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="STKVIN" >
                <xsd:simpleType>
                    <xsd:annotation>
                        <xsd:appinfo>
                            <ims:field name="STKVIN" alias="STKVIN"/>
                        </xsd:appinfo>
                    </xsd:annotation>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="20"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
            <xsd:element name="COLOR" >
                <xsd:simpleType>
                    <xsd:annotation>
                        <xsd:appinfo>
                            <ims:field name="COLOR" alias="COLOR"/>
                        </xsd:appinfo>
                    </xsd:annotation>
                    <xsd:restriction base="xsd:string">
                        <xsd:maxLength value="10"/>
                    </xsd:restriction>
                </xsd:simpleType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>

```

```

        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="PRICE" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="PRICE" alias="PRICE"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="5"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="LOT" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="LOT" alias="LOT"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="10"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
<xsd:element name="WRNTY" >
    <xsd:simpleType>
        <xsd:annotation>
            <xsd:appinfo>
                <ims:field name="WRNTY" alias="WRNTY"/>
            </xsd:appinfo>
        </xsd:annotation>
        <xsd:restriction base="xsd:string">
            <xsd:maxLength value="1"/>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:element>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>

```

A.9 Enhanced Provider MPP template sample

Example A-9 shows an enhanced template for a WSDL file that includes three operations, as described in “IMS template generation” on page 223.

Example A-9 Enhanced Provider MPP template sample

```

*process macro not('¬') or('|') rules(nolaxdcl);
*process limits(name(100) extname(8) fixeddec(15,31) fixedbin(31,63));

```

```

*process system(IMS) display(std);
/*****
 * IBM Rational Developer for System z
 * Enterprise Service Tools (EST)
 * Single-service Projects (XSE)
 * IMS Enterprise Suite SOAP Gateway
 * Web Service Provider Template MPP
 *
 * Date created: 2013-05-17 11:31:39 PDT
 * UUID: be97872d-c773-4e42-b9bd-f8ac4d45c450
 * INSTALLATION: 8.3.100
 *****/
WSPC1Package: package exports(*);
/* IMS PL/I service provider include */
%INCLUDE IRZPSH;
/* WSDL2ELS language structures */
%INCLUDE WSPC1;
/*-----*
 * Procedure: WSPC1
 * Description: Implementation of a Web service provider MPP for ser
 * vice "WS_poc_1" and binding "WS_poc_1_Binding" of the Web service
 * definition file "WS_PoC_1.wsdl".
 *-----*/
WSPC1: procedure(iopcb_mask_ptr) options(main);
dcl iopcb_mask_ptr pointer;
dcl host_text_1 char(1024) varying init('');
dcl host_text_2 char(1024) varying init('');

/* Begin mainline logic.
 */
@irz_iopcb_mask_ptr = iopcb_mask_ptr;
@irz_ee_feedback_ptr = addr(@irz_ee_feedback);
call ProcessMessages();
/* End mainline logic. */
return;
/*-----*
 * Procedure: ProcessMessages
 * Description: Retrieve the IRZPSIO message header from the IMS Me
 * ssage Queue and invoke a handler procedure based on the service c
 * ontext.
 *-----*/
ProcessMessages: procedure internal;
/* Allocate memory for the IRZPSIO message header and copy it fro
m the IMS Message Queue. */
allocate @irz_async_msg_header set (@irz_async_msg_header_ptr);
call CEETDLI(@irz_dli_get_unique, @irz_iopcb_mask,
 @irz_async_msg_header);
/* Branch to a handler procedure corresponding to the operation ci
ted in the service context. */
do while(@irz_iopcb_mask.iopcb_status_code =
 @irz_dli_status_ok);
select(@irz_async_msg_header.service_name);
when('WS_poc_1') do;
select(@irz_async_msg_header.operation_name);
when('getteam_1_0') do;

```

```

        call getteam_1_0Handler;
    end;
    when('setteam_1_0') do;
        call setteam_1_0Handler;
    end;
    when('ping_1_0') do;
        call ping_1_0Handler;
    end;
    otherwise do;
        host_text_1 = @irz_async_msg_header.operation_name;
        host_text_2 = @irz_async_msg_header.service_name;
        call LogEvent(trim(packagename()) || '#'
            || trim(procedurename()) || '(): Error, '
            || 'operation name "' || trim(host_text_1) || '" '
            || 'is not valid for service name "'
            || trim(host_text_2) || '".');
    end;
end;
end;
otherwise do;
    host_text_1 = @irz_async_msg_header.service_name;
    call LogEvent(trim(packagename()) || '#'
        || trim(procedurename()) || '(): Error, '
        || 'service name "' || trim(host_text_1) || '" '
        || 'is not valid or is not implemented by this '
        || 'application.');
```

end;

end;

call CEETDLI(@irz_dli_get_unique, @irz_iopcb_mask, @irz_async_msg_header);

end;

if (@irz_iopcb_mask.iopcb_status_code = @irz_dli_end_messages) then do;

host_text_1 = @irz_iopcb_mask.iopcb_status_code;

call LogEvent(trim(packagename()) || '#'

|| trim(procedurename()) || '(): Error, '

|| 'CEETDLI.GU(@irz_async_msg_header), '

|| 'iopcb_status_code: '

|| trim(host_text_1) || '.');

end;

free @irz_async_msg_header;

return;

end ProcessMessages;

/*-----*

* Procedure: getteam_1_0Handler

* Description: Get the request language structure for operation "ge

* tteam_1_0" from the IMS Message Queue using the IRZQGETS API, inv

* oke user-written implementation, set the response or fault langua

* ge structure and commit it to the IMS Message Queue using the IRZ

* QSETS API. Finally, deallocate language structures.

/*-----*/

getteam_1_0Handler: procedure internal;

/* Get the request language structure from the IMS Message Queue u

sing the IRZQGETS API.

*/

```

    @irz_struct_name = 'getteam_1_0';
    @irz_struct_type = @irz_soap_body_struct;
    @irz_struct_ptr = null();
    @irz_struct_size = 0;
    @irz_debug = '0'b;
call IRZQGETSWrapper(procedurename());
    if (@return_code = @irz_success) then do;
        getteam_1_0_ptr = @irz_struct_ptr;
    end; else do;
        return;
    end;
    /* Invoke user-written implementation of operation "getteam_1_0" passing
    pointers to the request, response, and fault language structures. */
    getteam_1_0Response_ptr = null();
    ServiceException_ptr = null();
    call getteam_1_0Impl(@irz_iopcb_mask_ptr, getteam_1_0_ptr, getteam_1_
0Response_ptr, ServiceException_ptr);
    /* Set the response or fault language structure for operation "get
    team_1_0" and commit it to the IMS Message Queue using the IRZQSET
    S API. */
    if (getteam_1_0Response_ptr != null()) then do;
        @irz_struct_name = 'getteam_1_0Response';
        @irz_struct_type = @irz_soap_body_struct;
        @irz_struct_ptr = getteam_1_0Response_ptr;
        @irz_struct_size = storage(getteam_1_0Response);
    end; else if (ServiceException_ptr != null()) then do;
        @irz_struct_name = 'ServiceException';
        @irz_struct_type = @irz_soap_fault_struct;
        @irz_struct_ptr = ServiceException_ptr;
        @irz_struct_size = storage(ServiceException);
    end;
    @irz_commit_structs = '1'b;
    @irz_debug = '0'b;

call IRZQSETSWrapper(procedurename());

    /* Deallocate language structures.
    */
    if (getteam_1_0_ptr != null()) then do;
        call plifree(getteam_1_0_ptr);
        getteam_1_0_ptr = null();
    end;
    if (getteam_1_0Response_ptr != null()) then do;
        call plifree(getteam_1_0Response_ptr);
        getteam_1_0Response_ptr = null();
    end;
    if (ServiceException_ptr != null()) then do;
        call plifree(ServiceException_ptr);
        ServiceException_ptr = null();
    end;

    return;

end getteam_1_0Handler;

```



```

/*-----*
* Procedure: getteam_1_0Impl
* Description: Implement the business logic for operation "getteam_
* 1_0" using the following steps: (1) Process the request language
* structure "getteam_1_0", (2) Allocate and fill in the response la
* nguage structure "getteam_1_0Response" or a fault language struct
* ure from the set {ServiceException}. (3) Finally, set the respect
* ive response or fault language structure pointer before returning
* to the operation handler (Handler) procedure. Note: If an error
* occurs that requires a rollback, use parameter iopcb_mask_ptr to
* access the IOPCB.
*-----*/
getteam_1_0Impl: procedure(iopcb_mask_ptr, getteam_1_0_ptr, getteam_1_0
Response_ptr, ServiceException_ptr) internal;

    dcl iopcb_mask_ptr pointer byvalue;
    dcl getteam_1_0_ptr pointer byvalue;
    dcl getteam_1_0Response_ptr pointer byaddr;
    dcl ServiceException_ptr pointer byaddr;

    return;

end getteam_1_0Impl;

/*-----*
* Procedure: setteam_1_0Handler
* Description: Get the request language structure for operation "se
* tteam_1_0" from the IMS Message Queue using the IRZQGETS API, inv
* oke user-written implementation, set the response or fault langua
* ge structure and commit it to the IMS Message Queue using the IRZ
* QSETS API. Finally, deallocate language structures.
*-----*/
setteam_1_0Handler: procedure internal;

    /* Get the request language structure from the IMS Message Queue u
    sing the IRZQGETS API.
    */
    @irz_struct_name = 'setteam_1_0';
    @irz_struct_type = @irz_soap_body_struct;
    @irz_struct_ptr = null();
    @irz_struct_size = 0;
    @irz_debug      = '0'b;

    call IRZQGETSWrapper(procedurename());
    if (@return_code = @irz_success) then do;
        setteam_1_0_ptr = @irz_struct_ptr;
    end; else do;
        return;
    end;

    /* Invoke user-written implementation of operation "setteam_1_0" p
    assing pointers to the request, response, and fault language struc
    tures.
    */
    setteam_1_0Response_ptr = null();

```

```

ServiceException_ptr = null();
call setteam_1_OImpl(@irz_iopcb_mask_ptr, setteam_1_0_ptr, setteam_1_
OResponse_ptr, ServiceException_ptr);

/* Set the response or fault language structure for operation "set
team_1_0" and commit it to the IMS Message Queue using the IRZQSET
S API.
*/
if (setteam_1_OResponse_ptr /= null()) then do;
    @irz_struct_name = 'setteam_1_OResponse';
    @irz_struct_type = @irz_soap_body_struct;
    @irz_struct_ptr = setteam_1_OResponse_ptr;
    @irz_struct_size = storage(setteam_1_OResponse);
end; else if (ServiceException_ptr /= null()) then do;
    @irz_struct_name = 'ServiceException';
    @irz_struct_type = @irz_soap_fault_struct;
    @irz_struct_ptr = ServiceException_ptr;
    @irz_struct_size = storage(ServiceException);
end;

@irz_commit_structs = '1'b;
@irz_debug = '0'b;

call IRZQSETSWrapper(procedurename());

/* Deallocate language structures.
*/
if (setteam_1_0_ptr /= null()) then do;
    call plifree(setteam_1_0_ptr);
    setteam_1_0_ptr = null();
end;
if (setteam_1_OResponse_ptr /= null()) then do;
    call plifree(setteam_1_OResponse_ptr);
    setteam_1_OResponse_ptr = null();
end;
if (ServiceException_ptr /= null()) then do;
    call plifree(ServiceException_ptr);
    ServiceException_ptr = null();
end;

return;

end setteam_1_0Handler;

/*-----*
* Procedure: setteam_1_OImpl
* Description: Implement the business logic for operation "setteam_
* 1_0" using the following steps: (1) Process the request language
* structure "setteam_1_0", (2) Allocate and fill in the response la
* nguage structure "setteam_1_OResponse" or a fault language struct
* ure from the set {ServiceException}. (3) Finally, set the respect
* ive response or fault language structure pointer before returning
* to the operation handler (Handler) procedure. Note: If an error
* occurs that requires a rollback, use parameter iopcb_mask_ptr to
* access the IOPCB.

```

```

/*-----*/
setteam_1_0Impl: procedure(iopcb_mask_ptr, setteam_1_0_ptr, setteam_1_0
Response_ptr, ServiceException_ptr) internal;

    dcl iopcb_mask_ptr pointer byvalue;
    dcl setteam_1_0_ptr pointer byvalue;
    dcl setteam_1_0Response_ptr pointer byaddr;
    dcl ServiceException_ptr pointer byaddr;

    return;

end setteam_1_0Impl;

/*-----*
* Procedure: ping_1_0Handler
* Description: Get the request language structure for operation "pi
* ng_1_0" from the IMS Message Queue using the IRZQGETS API, invoke
* user-written implementation, set the response or fault language
* structure and commit it to the IMS Message Queue using the IRZQSE
* TS API. Finally, deallocate language structures.
*-----*/
ping_1_0Handler: procedure internal;

    /* No request language structure is defined for operation "ping_1_
    0".
    */

    /* Invoke user-written implementation of operation "ping_1_0" pass
    ing pointers to the request, response, and fault language structur
    es.
    */
    call ping_1_0Impl(@irz_iopcb_mask_ptr);

    /* No response language structure is defined for operation "ping_1
    _0". Insert only the IRZPWSIO message header.
    */
    call CEETDLI(@irz_dli_insert, @irz_iopcb_mask,
        @irz_async_msg_header);

    return;

end ping_1_0Handler;

/*-----*
* Procedure: ping_1_0Impl
* Description: Implement the business logic for operation "ping_1_0
* " using the following steps: (1) Process the request language str
* ucture "n/a", (2) Allocate and fill in the response language stru
* cture "n/a" or a fault language structure from the set {}. (3) Fi
* nally, set the respective response or fault language structure po
* inter before returning to the operation handler (Handler) procedu
* re. Note: If an error occurs that requires a rollback, use parame
* ter iopcb_mask_ptr to access the IOPCB.
*-----*/
ping_1_0Impl: procedure(iopcb_mask_ptr) internal;

```

```

    dcl iopcb_mask_ptr pointer byvalue;

    return;

end ping_1_0Impl;

/*-----*
 * Procedure: IRZQGETSWrapper
 * Description: Get a language structure from the IMS Message Queue
 * using the IRZQGETS API.
 *-----*/
IRZQGETSWrapper: procedure(caller_proc_name) internal;

    dcl caller_proc_name char(*) byaddr;

    @return_code = IRZQGETS(@irz_async_msg_header_ptr,
        @irz_iopcb_mask_ptr, @irz_struct_type,
        @irz_struct_name, @irz_struct_ptr,
        @irz_struct_size, @irz_cee_feedback_ptr,
        @irz_debug);

    if (@return_code /= @irz_success) then do;
        call LogEvent(trim(packagename()) || '#'
            || trim(caller_proc_name) || '(): '
            || 'Error, IRZQGETS return code: '
            || trim(@return_code) || '.');
        if (@irz_struct_ptr /= null()) then do;
            call plifree(@irz_struct_ptr);
            @irz_struct_ptr = null();
            @irz_struct_size = 0;
        end;
    end;

    return;

end IRZQGETSWrapper;

/*-----*
 * Procedure: IRZQSETSWrapper
 * Description: Set a language structure and optionally commit it to
 * the IMS Message Queue using the IRZQSETS API.
 *-----*/
IRZQSETSWrapper: procedure(caller_proc_name) internal;

    dcl caller_proc_name char(*) byaddr;

    @return_code = IRZQSETS(@irz_async_msg_header_ptr,
        @irz_iopcb_mask_ptr, @irz_struct_type,
        @irz_struct_name, @irz_struct_ptr,
        @irz_struct_size, @irz_commit_structs,
        @irz_cee_feedback_ptr, @irz_debug);

    if (@return_code /= @irz_success) then do;
        call LogEvent(trim(packagename()) || '#'

```

```

        || trim(caller_proc_name) || '(): '
        || 'Error, IRZQSETS return code: '
        || trim(@return_code) || '.';
end;

return;

end IRZQSETSWrapper;

/*-----*
 * Procedure: LogEvent
 * Description: Common procedure for logging events that may be exte
 * nded as needed.
 *-----*/
LogEvent: procedure(text) internal;
    dcl text char(*) varying byaddr;
    put skip list(datetime() || ' ' || text);
    put list('');
    return;
end LogEvent;

end WSPOC1;

end WSPOC1Package;

```



Additional material

This book refers to additional material that can be downloaded from the Internet as described in the following sections.

Locating the web material

The web material that is associated with this book is available in softcopy on the Internet from the IBM Redbooks web server. Point your web browser at:

<ftp://www.redbooks.ibm.com/redbooks/SG248174>

Alternatively, you can go to the IBM Redbooks website at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redbooks form number, SG248174.

Using the web material

The additional web material that accompanies this book includes the `sg248174.zip` file, which includes the following file:

<i>File name</i>	<i>Description</i>
SQL12I.txt	COBOL native dynamic SQL program

System requirements for downloading the web material

The web material requires the following system configuration:

Hard disk space:	2 MB minimum
Operating System:	Windows
Processor:	Intel 386 or higher
Memory:	16 MB

Downloading and extracting the web material

Create a subdirectory (folder) on your workstation, and extract the contents of the web material compressed file into this folder.

Related publications

The publications that are listed in this section are considered suitable for a more detailed description of the topics that are covered in this book.

IBM Redbooks

The following IBM Redbooks publications provide more information about the topics in this document. Some publications that are referenced in this list might be available in softcopy only.

- ▶ *IBM IMS Version 10 Implementation Guide: A Technical Overview*, SG24-7526
- ▶ *IBM IMS Version 12 Technical Overview*, SG24-7972
- ▶ *IMS 12: The IMS Catalog*, REDP-4812
- ▶ *IMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity*, SG24-6794
- ▶ *IMS e-business Connectors: A Guide to IMS Connectivity*, SG24-6514
- ▶ *IMS Version 11 Technical Overview*, SG24-7807
- ▶ *Powering SOA Solutions with IMS*, SG24-7662
- ▶ *Rethink Your Mainframe Applications: Reasons and Approaches for Extension, Transformation, and Growth*, REDP-4938

You can search for, view, download, or order these documents and other Redbooks, Redpapers, Web Docs, draft and more materials, at the following website:

ibm.com/redbooks

Other publications

These publications are also relevant as further information sources:

- ▶ *IBM IMS Connect Extensions for z/OS Version 2 Release 2, User's Guide*, SC19-2817
- ▶ *IBM IMS Performance Analyzer for z/OS Version 4 Release 2 User's Guide*, SC19-2732-01
- ▶ *IMS TM Resource Adapter User's Guide and Reference*, SC19-1211-02
- ▶ *IMS V13 Communications and Connections*, SC19-3651-00
- ▶ *IMS V13 Release Planning*, GC19-3658
- ▶ *IMS Version 13 Commands, Volume 1: IMS Commands A-M*, SC19-3648-00
- ▶ *IMS Version 13 Commands, Volume 2: IMS Commands N-V*, SC19-3649-00
- ▶ *IMS Version 13 Commands, Volume 3: IMS Component and z/OS Commands*, SC19-3650-00
- ▶ *Security Server RACF Security Administrator's Guide*, SA22-7683-14

- ▶ *z/OS V1R12.0 UNIX System Services Planning*, GA22-7800-18
- ▶ *z/OS V1R13 Communications Server IP Configuration Guide*, SC31-8775-20

Online resources

These websites are also relevant as further information sources:

- ▶ IBM WebSphere DataPower SOA Appliances
<http://www.ibm.com/software/integration/datapower/index.html>
- ▶ *IMS Version 12 Performance Evaluation Summary - Maintaining superior performance*, found at:
http://public.dhe.ibm.com/software/data/sw-library/ims/IMS_Version_12_Performance_Summary.pdf
- ▶ *IMS 13 Performance Evaluation Summary - Reducing the Total Cost of Ownership with Improved Performance*, found at:
<http://public.dhe.ibm.com/common/ssi/ecm/en/im114383usen/IML14383USEN.PDF>
- ▶ WebSphere Application Server
<http://pic.dhe.ibm.com/infocenter/wasinfo/v8r5/index.jsp>

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Symbols

.NET 368
.NET Framework 368

A

A047 record 107
ACCEPT 14, 141, 179
ACEE 32, 86, 152
ACK 10, 12, 101, 133, 169, 197–198, 231
ACK/NAK 12, 101, 133
ActivationSpec 244, 248, 438
ADAPTER 18
address space 7, 87, 162
ADO.NET 368
AIB 222, 299–300, 415, 431, 433
aib 245, 300
AIBREASN 300, 431
AIBRETRN 300, 431
ALTPCB 64, 69, 144, 148, 201, 239–241
APF 14–15
APIs 4, 6, 224, 254
APPL 20
application program 12, 157, 275, 354, 401, 415
application programming
 interface 62, 250, 397
APPLY 14, 179
ASCII 356
Asynchronous callout 238, 240
asynchronous callout 157, 159, 241–242, 437
asynchronous hold queue
 name 201, 243
 SGTP1 TPIPE 201, 203
asynchronous output 20, 74, 144, 150–151, 225, 231, 238
ATF 110
AT-TLS 56

B

BPE 14–15, 186
BPECFG 15
BPECFGxx 14
BPXPRMxx 16
business process 5

C

callout EJB 243
callout message 160, 194, 239, 242, 416, 436–437
 OTMA headers 201
 XML transformation 206
callout request 171, 228, 253, 412–413, 415
callout request message 242
 Web Service identifier 199

CICS connections 257
client application 13–14, 94, 167, 174, 226, 231, 412
client bid 75, 129
clientID 238
CLOSE 264
CLOSEHWS 43, 45
COBOL 61, 139, 141–142, 169–170, 218, 274, 280, 348–349, 409, 430
COBOL copybook 173, 189, 226, 275, 325
cold start
 IMS 284
command line
 interface 399
CommandBuilder 377
commit mode 10–11, 101, 136, 166, 226, 230, 436, 441
commitMode 230
commit-then-send 10–11, 132, 149, 230
Components of SOAP Gateway 178
configuration member 15, 229
Connection bundle
 Property options 171–172
connection bundle 171
 file 171, 173
Connection Factory 235–236, 436, 441
connection pool 373
connection pooling 5, 225, 227, 235, 373
ConnectionFactory 241, 435
conversational 131, 235, 237
conversations 12, 135, 148, 249
converter name 172, 200
Correlator file 172
correlator file 172
Coupling Facility 127, 131
CSL 7, 255
CSLDCxxx member 422
CSQZPARM 128
CSSLIB 15

D

data sharing xxv, 289
data transformation 173, 202, 392, 395–396, 398
database xxv, 9, 12, 30, 85, 152, 200, 215, 228, 230, 273, 349, 354, 368, 388, 395, 400, 421–422
 IMS 12, 86, 240, 273, 354, 360, 368, 388, 400, 422
DataPower 5, 37, 384, 395
DataPower and IMS Integration 401
DataPower appliance 398, 413
datastore 9, 91, 94, 192, 440
DATASTORE 19
datastore name 19
DATASTORE statement 19, 414–415
 ID parameter 65
DB2 11, 88, 128–129, 222, 369, 396
DB2 for z/OS 264

DBCTL 30, 109, 212
 DBRC 281
 DCCTL 9
 DECLARE CURSOR 264
 DECLARE STATEMENT 264
 DEDB 360
 DELETE 31, 135, 264, 375, 415, 435
 Demand Environment 37
 Demilitarized Zone 396
 dependent region 133, 162, 200, 207, 235, 252, 401, 415
 DESCRIBE OUTPUT 265
 descriptor DESTSG1 201
 destination name 158
 DFS058I 142, 224
 DFS555I 167
 DFSCCMD0 152
 DFSCTRN0 152
 DFSDDLTO 299
 DFSIVP37 354, 358
 DFSPBxxx 19, 131
 DFSYDRU0 20, 37, 64, 128, 151, 156, 239, 241
 DFSYDTx 128, 133, 194, 239, 414, 418
 DFSYPRX0 156–157, 159, 239, 241
 Disconnected Mode 376
 DL/I 109, 167, 207, 222, 274–275, 298, 365, 414–416
 DL/I call 110, 222, 293, 298
 DLIModel 305, 349, 445–446
 DRU 20, 132, 137
 DSN 15, 140, 282, 415, 430
 DSNAIMS 129
 duplicate client 105
 dynamic allocation 289
 dynamic SQL 264
 dynamic VIPA 39

E

EBCDIC 175, 294, 296, 356, 381, 411
 ECB 25
 EJB 208, 215–216, 435, 440
 Enterprise COBOL Version 5, 264
 ESS interface 128
 EXECUTE 265
 exit 20–21, 99, 128, 131–132, 185, 213, 240–241, 281, 380, 404, 411–412, 414
 exit routine 21, 151, 412, 415
 external application 199, 207, 240, 253, 401
 Send-Only message processing functions 208

F

FACILITY class 30, 151
 factory-based 371
 failover 39, 228
 Fast Path 131, 354, 359
 feed 443
 FETCH 265
 FMID 175, 179

G

generic coding 371
 GRNAME 19, 131
 GROUP 19
 GSAM 285

H

HALDB 275
 HIDAM 301
 hold queue 34, 74, 133, 153, 231, 239
 callout request 239, 241
 output messages 154
 host name 171, 229, 411, 417
 HOSTNAME 26, 28, 139
 HTTP 171, 179, 216, 385, 399, 407
 HWS 15, 18, 139, 403, 414, 421
 HWS statement 20
 HWSCFGxx 14, 37–38, 213
 HWSCSLO0 26, 37, 139–140
 HWSCSLO1 26, 37, 139–140
 HWS0252W 142
 HWSIMSO0 36, 139–140
 HWSIMSO1 36, 139
 HWSJAVA0 17, 26
 HWSRCORD 15
 HWSSMPL0 17, 21, 28, 139–140, 255
 HWSSMPL1 17, 21, 36, 139–140, 255
 HWSUINIT 21
 HWSYDRU0 20, 28, 139

I

IBM IMS xxv, 77, 85, 224, 273, 476
 IBM IMS Data Provider
 programming applications 371
 IBM IMS Data Provider for Microsoft .NET, 367
 IBM MobileFirst 383
 IBM Worklight 385, 396
 ICAL call 194, 246, 415, 433
 ICAL SENDRECV
 call 207, 245
 synchronous call function 207, 245
 ID 19
 IMD Data Provider
 result sets 376
 IMS xxv, 1, 3, 7, 92–93, 127, 169, 217, 252, 261, 273, 347, 366–367, 395–396, 430, 435–436
 IMS application 34, 64, 128, 149, 170, 223–224, 302, 304, 347, 349, 392, 398, 402–403, 443
 alternate TPPCB 199
 App1 201–202
 App2 204, 206
 Appl2 200, 204
 callout processing 212
 data format 200
 data structure 175
 format response 174
 format transformation 174
 instance 199, 207
 issue 212, 246

- program 149, 188, 207, 231, 304, 354
- programming style 207
- IMS command 32, 55, 129, 135, 224–225
- IMS Connect 4, 7, 85, 91–93, 128, 170–171, 224, 384, 402–403, 440
- IMS Connect Base Primitive Environment 17
- IMS Connect BPE 17
- IMS Connect commands 43
- IMS Connect Dump Formatter 81
- IMS Connect Extensions 40, 62, 85, 91
- IMS Connect security 30
- IMS Connect SSL 76
- IMS Connectivity 37
- IMS Connector for Java 26, 31, 224, 237
- IMS Control Center 7, 26, 37, 135
- IMS conversation 249
- IMS conversational transaction 237
- IMS data xxvi, 6, 63, 87, 228, 275, 347, 349, 388, 416, 418
- IMS Data Provider
 - API 370
 - architecture 369
 - prerequisites 369
 - processing metadata 377
 - tracing 380
- IMS database 63, 294, 360, 388, 422
- IMS DB 7, 222, 350, 401, 421
 - TCP/IP path 63
- IMS Monitor Trace 48
- IMS Open Database 30, 87, 273, 365
- IMS Open Database Manager 62
- IMS OTMA 9, 19, 128, 186, 224, 229, 403, 413, 418
 - Asynchronous Queue 238
 - group 19, 130
 - member name 131
- IMS Performance Analyzer 89, 91–92
- IMS Problem Investigator 88, 109, 167
- IMS SOAP Gateway 5, 62, 169, 244, 367, 385, 414
 - 10.1 207, 212
 - 10.2 212
 - Asynchronous Callout 199
 - feature 207
 - message 174
 - Multiple segment support 214
 - RESUME TPIPE request processing 202
 - second Web Service 202
 - server 173, 199
 - solution 170
 - tool 171
 - Version 199
 - Web Services 199
 - XML response message 200
- IMS subsystem 14
- IMS system 7, 9, 116, 134, 136, 200, 224, 276, 283, 403, 413–414
- IMS TM 4, 6–7, 157, 207, 222, 224, 348, 401
- IMS TM Resource Adapter 7, 208, 224
- IMS Transaction
 - Manager 224–225
- IMS transaction 32, 69, 90, 128–129, 172, 175, 224, 228,

- 391
 - code 129, 175, 204
 - data 149, 175, 204, 244
- IMS Universal Database resource adapters 256
- IMS Universal DL/I driver 256
- IMS Universal drivers 256
- IMS Universal JDBC Driver 388
- IMS Universal JDBC driver 256, 422
- IMS Version
 - 10 14, 130, 199, 224, 228
 - 10 Implementation Guide 31
 - 10 SPE 246
 - 11 9, 30, 130, 224
 - 9 9
- IMS Version 11 7, 133, 214
- IMSCON TMEBER 414
- IMSConnectionFactory 250
- IMSConnectionSpec 241, 250, 333, 435–436
- IMSDataReader.GetSchemaTable method 378
- IMSID 89, 284
- IMSInteractionSpec 225, 435
- IMSMON 48
- IMSPParameter 373
- IMSPLEX 9, 21, 139–140
- IMSplex 7, 9, 163, 290–291
- IMSPLEX statement 19
- IMSTransaction object 375
- IMSXCF.group.member 30
- INCLUDE 265
- InfoSphere MashupHub 443
- input message 32, 34, 97, 132
- input parameter 253
- INSERT 265
- INSTALL 403
- INSTALL/IVP 413
- installing 176, 233
- integrated IMS 175
- integrated IMS Connect function 31, 252
- InteractionSpec 237, 243, 441
- Internet 14, 16, 301
- Internet Protocol
 - Version 6 26
- IOPCB 128, 149, 228, 230, 433
- IP address 14, 39, 99, 210, 331–332, 362, 411, 417, 423
- IPCS 81
- IPV6 14, 16, 139
- ISC 7
- ISRT ALTPCB 242, 244
 - call 244
- IVP 252, 403, 413
- IVTNO 92

J

- J2C 224–225
- J2C Java bean 226
- J2EE 5, 215–216
- J2EE Connector architecture 216
- Java 4, 26, 30, 169–170, 215–216, 275, 349, 360, 371, 387, 440
- Java application 74, 216–217, 294, 297, 389

- Java applications 5–6, 216, 224, 305, 330
- Java development 216
- Java drivers 63
- Java language 294
- java.io 332
- JBP 207, 212, 252, 275
- JCA 4–5, 225
- JCL 14–15, 181, 221, 282, 349
- JDBC 30, 275, 305, 349, 361–362, 388, 401, 422
- JDBC driver 334, 422
- JMP 207, 252, 275
- JSON 396
- JVM 198

K

- keystore 196, 229

L

- LANG 17, 353
- Language Environment 25
- Linux 7, 170, 182, 221, 225–226
- load balancing 62, 392, 396
- Local Option 249
- logical relationships 274
- logrecord 187
- LPAR 5, 35, 127, 179, 221, 332, 398

M

- MAXFILEPROC 26
- MAXSOC 26, 29
- MAXSOCKETS 16
- MDB 208, 228, 437–438
- MEMBER 19
- metadata 178, 273, 349, 360, 389
- MFS 73, 221, 226
- middleware 1, 6, 227
- migration 38, 169, 251, 291
- MOD 73, 430
- mode 10–11, 101, 131, 137, 169, 176–177, 252–254, 286, 376, 400, 419, 436, 441, 447
- mode 0 10, 136, 166, 226
- MPP 194, 207, 223, 303, 451–452
- MQSeries 134, 159
- MSC 7, 22, 131
- MSDB 360
- multiple IMS
 - application 209, 274
 - system 9
- MVS 33, 47, 86, 127, 176–177, 222, 289, 398

N

- NAK 10, 12, 92–93, 101, 133, 200
- NAK response 33, 163
- namespace 445
- Native SQL support for COBOL 263
- non-persistent socket 13

O

- ODACCESS 23
- OM 7, 9, 96, 134, 255
- OMEGAMON for IMS on z/OS 110
- online change 288
- OPEN 265
- Open Database
 - Manager 7, 422
- Open Transaction Manager Access 9, 128
- OPENDS 43, 47
- OPENIP 43, 48
- OPENPORT 40, 43
- operational controls 96
- Operations 7, 86
- operations 42, 56, 86, 130–131, 173, 176, 184, 224, 334, 375, 399, 416
- Operations Manager 9, 21, 134
- OTMA 7, 9, 86, 92–93, 127, 185–186, 224, 253, 403, 430, 433–434
 - descriptors 135, 239, 414
 - transaction pipe 147
- OTMA C/I 129
- OTMA callable interface 130
- OTMA callout security 153
- OTMA client 11, 128, 131, 242, 244, 247
 - associates application output 147
 - descriptor 137
 - front-end IMS 145
 - security levels 155
- OTMA descriptor 33, 133, 140, 200, 208, 239, 254
 - name 64, 208, 246
- OTMA Destination
 - Resolution exit routine 37, 241
- OTMA flood condition 114
- OTMA header 151
- OTMA message format 9
- OTMA protocol
 - command 137
- OTMA security 151–152, 241
- OTMA super member 148
- OTMAASY 131
- OTMACON 128
- OTMAMD 131
- OTMANM 20, 131
- OTMASE 131
- OTMASP 131
- output message 10, 102, 136, 148, 174–175, 228, 231
- output XML 174

P

- Parallel xxv
- Parallel Sysplex xxvi
- PCB 164, 222, 231, 274–275, 353
- performance xxv, 3, 10, 14, 86, 92–93, 127, 184–185, 254, 342, 401, 424
- persistent socket 13, 116, 230
- PL/I 175, 208, 221, 223, 274–275, 348–349, 409, 452
- PM88737 165
- PM92523 263

- port 14, 26–27, 91, 164, 171, 184, 229, 238, 331–332, 362, 372, 386, 400, 404, 407, 411, 446
- PORTID 27, 139, 404, 415
- POSIX 25
- PPT 14
- Practical Guide 37
- PREPARE 265
- Prepare method 373
- PROCLIB 15, 17, 131, 133, 135, 194, 239, 282–283, 403, 414, 418
- program communication
 - block 274, 354
- programming model 170, 207, 230, 376
- program-to-program switch 131
- PROGxx 15
- project xxvi–xxvii, 216, 306, 311, 363, 388
- PSBGEN 275, 353
- PTKTDATA statement 20
- public class
 - MDB4IMASyncCallOut1Bean 437
 - MDB4IMSSyncCallOut1Bean 442
 - SLSB4IMASyncCallOut1Bean 435
 - SLSB4IMSSyncCallOut1Bean 440

Q

- QUERY 32, 118, 135
- QUERY MEMBER 33, 54

R

- RACF 20, 92, 131, 134, 139, 213, 232, 412
- RACF FACILITY class 75, 153
- RACFID 27
- RAD 232, 349
- RAR 216, 225
- RATE 92–93
- Rational Application Developer 225–226
- Rational Developer 120, 175, 215, 221, 307–308, 452
- Rational Software
 - Architect 225
- RDz 61–62, 172–173, 215, 349, 356
- RECEIVE 14, 179, 410
- RECORDER 18, 43
- Request Mod Message 73
- Resource 7, 12, 157, 163, 208, 216, 224, 276, 439
- resource 4, 6, 12–13, 86, 91, 134, 141, 224, 288, 299, 344, 435
- Resource Adapter 31, 38, 224, 443
- response message 74, 141–142, 188, 253–254, 437–438
- restart
 - IMS 9, 131, 194, 249, 403
- RESUME TPIPE 202–203, 241–242, 244
 - request processing 203, 205
- Resume TPIPE 199, 208, 240, 246
- Resume Tpipe
 - request 74, 148
- RMTICIS 24
- RMTIMSCON 25
- RRNAME 20

- RRS 12, 87, 128, 133
- RUNOPTS 25

S

- SAF 65, 75, 92–93, 152–153, 229, 418
- same IMS 9, 155, 207, 228, 238, 330, 416
 - transaction instance 208, 210
- sample IMS COBOL SQL program 265
- SCEERUN 15, 430
- scheduling 10, 131, 244, 248, 303
- SCHEDxx 16
- SCI 9
- SDFSMAc 31, 147
- SDFSRESL 15, 282
- SDFSsrc 280
- Security 37, 56, 87, 151–152, 175, 178, 235, 240, 390, 400
- security 4–5, 14–15, 86, 92, 131, 151, 169, 171, 173, 225, 229–230, 368, 384, 395–396
- security data 75
- SECURITY macro 154
- SELECT 265
- send-then-commit 11, 132–133, 230
- service data 202
- Session rebalancing 118
- SETRACF 20, 43, 49
- SETRRS 43, 49
- Shared Queues 145, 148, 210–211, 246
- shared queues xxv–xxvi, 145, 293
- SMB 151
- SMP/E 14, 176, 251
- SNA terminal 157
- SOAP 4, 17–18, 44, 157, 169, 224, 367, 385, 400, 452
- SOAP Gateway 4–5, 36, 62, 169, 244, 385, 414
 - Multi-segment handling 212
- SOAP message 174, 195, 200
- sockets 13–14, 226
- SPA 249
- SPOC 9, 42, 135
- SQL 87, 129, 304, 349, 355, 385, 397, 399, 401
- SSL port 27
- SSL support 15, 171
- SSL Truststore
 - name 171
- SSLENVAR 27, 44, 139
- SSLPORT 27, 44, 139
- STARTDS 43, 47
- STARTIP 43, 48
- STARTP 48
- STARTPT 43, 54
- state data 164, 416
- STGCLASS 129
- STOPCLNT 43, 50
- STOPDS 43, 50
- STOPIP 43, 50
- STOPPORT 43, 48
- storage 17, 33, 129, 132, 249, 299, 360, 454
- stored procedure 129
- Structured Call Interface 9
- STSN 10

- super member 34, 139, 148
- sync level=confirm 12
- sync level=none 11
- sync point 11, 128
- synchronization level 11
- Synchronous callout 112, 149, 194, 196, 228, 232, 401, 416
- synchronous callout 34, 112, 149–150, 169, 252, 406, 412–413, 440–441
 - call 253, 414–415
 - function 207
 - message 194, 252–253, 416
 - request 150, 198, 207, 246, 415–416
 - solution 190, 414
 - support 150, 169, 189, 252, 413
- synchronous callout request
 - message 207–208, 246–247
 - status 208
- synchronous callout support 414
- SYS1.PARMLIB 14
- Sysplex xxv, 14, 118, 127, 179
- sysplex xxv, 34, 38–39, 86, 127, 210
- Sysplex Distributor 39–40, 210
- system
 - generation 65, 275
- system definition 119, 131, 403
- system generation 66
- System z 120, 170, 173, 220, 311, 347, 396, 452

T

- takeover 39
- TCP 7, 85, 142, 186, 229, 386, 412
- TCP/IP 4, 7, 86, 128, 174, 221, 229, 411, 417
- TCP/IP client 9, 254
 - application 73
- TCP/IP clients 9, 148, 254
- TCP/IP communication 65
- TCP/IP connection 35, 69, 202–203, 422
- TCP/IP port 50
- TCP/IP statement 25
- TIMEOUT 28, 92, 133, 434
- timeout 27, 103, 133, 230, 253–254, 361, 407, 412, 414, 436, 441
- TM Resource Adapter 38, 224
- TMEMBER 12, 20, 129, 132, 193, 203, 239, 414
- TMEMBER name 21, 201, 205
- TMEMBER 21
- TMRA 4, 225
- TPIPE 171, 193, 199, 228, 239, 414
- Tpipe 10, 129, 247
- tpipe 34, 144, 164, 229, 231, 416, 418
- TPIPE function 208
- TPIPE name 201, 203, 418
- tpipe name 164, 241–242, 418
- TRANSACT 92–93, 148
- transaction code 65, 77, 91–92, 105, 151, 172, 200
- transaction socket 74, 101
- TRCLEV 17
- truststore 196, 229
- TSO 54, 135, 182, 284, 349, 362, 423

- two-phase commit 11, 13, 225, 249
- type-1 command 131, 403, 413
- type-2 command 42

U

- UK82636 402
- UK91544 402
- UML 222
- Unicode 251, 412
- UNIX System Services 26, 176
- UOR 13, 55
- UPDATE 54, 135, 148, 265, 337, 375
- URL 361, 386, 409–410
- user data 39, 128, 185
- user exit 19–20, 27, 131, 153, 255
- user exits 15, 152
- user initialization exit 21
- user message exit 21, 26, 412
- Userid 89, 194

V

- VIEWDS 32–33
- VIEWHWS 32, 141–142, 362
- VIEWIP 43, 53
- VIEWPORT 43, 53
- VIEWUOR 55, 57
- VIPA 39
- virtualization 1
- VSAM 48, 222, 301
- VTAM 65, 127

W

- Web 2.0 391, 396
- Web Service 170, 244, 387, 452
 - one-way operation 201
 - request-response operation 204
 - secure manner 213
- Web service 5, 191, 194, 217, 392, 452
- Web Services 169, 171, 224–225, 386, 395, 445–446
 - Service 199, 228
- WebSphere 4–5, 37, 75, 120, 128, 178, 215, 224–225, 384, 395, 397–398, 435, 443
- WebSphere Application
 - Server 437, 442
- WebSphere Application Server 440
- WebSphere Application Server 5–6, 208, 211, 224–225, 385, 396–397, 439
- WebSphere Application Server Community Edition 249
- WebSphere DataPower SOA appliances 396
- WebSphere MQ 6, 128
- WebSphere MQ API 128
- WHENEVER 265
- WLM 39
- workload balancing 38–39
- Workload Manager 39
- WSDL 171, 217–218, 386, 400
- WSDL file 173, 218, 224
- WTO 132

X

- XCF 9, 127–128, 224, 229, 403, 413
- XCF group 19, 128–129
- XCF member 19, 131
- XCFGNAME 129
- XCFMNAME 129
- XIB 21
- XIBAREA 21, 28, 139
- XML 4–5, 15, 44, 158, 170, 172, 217–218, 275, 297, 299, 385, 396, 399
- XML adapter 200
- XML conversion 173
- XML converter 173
- XML data 174–175
 - structure 175
- XML document 173, 299
 - data 299
- XML message 173–174
 - format 175
- XML schema 401
- xml version 386, 420, 443, 445
- XQuery 363, 399

Z

- z/OS 5–7, 127, 169–170, 175, 221, 305–307, 349, 362, 390, 396, 403, 413
- z/OS UNIX System Services 26



IMS Integration and Connectivity Across the Enterprise

(1.0" spine)

0.875" x 1.498"

460 <-> 788 pages



IMS Integration and Connectivity Across the Enterprise

Modernize IMS batch and online applications with interoperability

Use the catalog for IMS database and application metadata information

Expand client access to IMS and DB2 data

This IBM Redbooks publication gives a broad understanding of IMS integration and connectivity solutions to access applications and data stores across your enterprise architecture.

As an application developer, architect, systems integrator, or systems programmer, there is important information that is available in this book that pertains to your responsibilities to continue to include the proven performance, data integrity, and workload distribution that is available from IMS in to selected projects that are related to your entire enterprise.

This book updates and adds to the information in the following IBM Redbooks publications:

- ▶ *IMS e-business Connectors: A Guide to IMS Connectivity*, SG24-6514
- ▶ *IMS Connectivity in an On Demand Environment: A Practical Guide to IMS Connectivity*, SG24-6794
- ▶ *Powering SOA Solutions with IMS*, SG24-7662
- ▶ *IBM IMS Version 12 Technical Overview*, SG24-7972
- ▶ *IMS 12: The IMS Catalog*, REDP-4812
- ▶ *Rethink Your Mainframe Applications: Reasons and Approaches for Extension, Transformation, and Growth*, REDP-4938

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

SG24-8174-00

ISBN 0738439010