

Implementation of
ISO/IEC 9579-1:1993

Information technology — Open Systems Interconnection — Remote Database Access —

Part 1: Generic Model, Service and
Protocol

UDC 681.3:621.39

NO COPYING WITHOUT BSI PERMISSION EXCEPT AS PERMITTED BY COPYRIGHT LAW

BSI
Documents

AVA

Document Sharing Hub

Committees responsible for this British Standard

The preparation of this British Standard was entrusted by the Information Systems Technology Standards Policy Committee (IST/-) to Technical Committee IST/21, upon which the following bodies were represented:

- British Computer Society
- British Telecommunications plc
- CCTA (the Government Centre for Information Systems)
- Department of Trade and Industry [IT Standards Unit (ITd6a)]
- Digital Equipment Co. Ltd.
- IBM United Kingdom Ltd.
- Information Systems Committee of the Universities Funding Council
- International Computers Limited
- Level 7 Ltd.
- Logica UK Ltd.
- Ministry of Defence
- National Computing Centre Ltd.
- National Health Services
- Nine Tiles Computer Systems Ltd.
- Oracle Corporation (UK) Ltd.
- PSC International Ltd.
- Rank Xerox Ltd.
- SEMA Group Systems Ltd.
- Salford Software Services (3-5)
- X Open Company Ltd.

This British Standard, having been prepared under the direction of the Information Systems Technology Standards Policy Committee, was published under the authority of the Standards Board and comes into effect on 15 April 1994

© BSI 05-2000

The following BSI references relate to the work on this standard:
Committee reference IST/21
Draft for comment 91/67852 DC

ISBN 0 580 23066 X

Amendments issued since publication

Amd. No.	Date	Comments

Contents

	Page
Committees responsible	Inside front cover
National foreword	ii
Foreword	xiii
Text of ISO/IEC 9579-1	1

National foreword

This British Standard reproduces verbatim ISO/IEC 9579-1:1993 and implements it as the UK national standard.

This British Standard is published under the direction of the Information Systems Technology Standards Policy Committee whose Technical Committee IST/21 has the responsibility to:

- aid enquirers to understand the text;
- present to the responsible international committee any enquiries on interpretation, or proposals for change, and keep UK interests informed;
- monitor related international and European developments and promulgate them in the UK.

NOTE International and European Standards, as well as overseas standards, are available from Customer Services, Publications, BSI, Linford Wood, Milton Keynes, MK14 6LE.

A British Standard does not purport to include all the necessary provisions of a contract. Users of British Standards are responsible for their correct application.

Compliance with a British Standard does not of itself confer immunity from legal obligations.

Summary of pages

This document comprises a front cover, an inside front cover, pages i and ii, the ISO/IEC title page, pages ii to xiv, pages 1 to 151 and a back cover.

This standard has been updated (see copyright date) and may have had amendments incorporated. This will be indicated in the amendment table on the inside front cover.

INTERNATIONAL
STANDARD

ISO/IEC
9579-1

First edition
1993-12-15

**Information technology — Open Systems
Interconnection — Remote Database
Access —**

Part 1:
Generic Model, Service and Protocol

*Technologies de l'information — Interconnexion de systèmes ouverts
(OSI) — Accès aux bases de données à distance —*

Partie 1: Modèle, service et protocole



Reference number
ISO/IEC 9579-1:1993(E)

Contents

	Page
Foreword	xiii
Introduction	xiv
Section 1. Introduction	
1.1 Scope	1
1.2 Normative references	3
1.3 Definitions	4
1.3.1 Basic Reference Model	4
1.3.2 Reference Model — Naming and Addressing	4
1.3.3 Service conventions	4
1.3.4 Application Layer Structure	5
1.3.5 Connection Oriented Presentation Service Definition	5
1.3.6 Service Definition for the Association Control Service Element	5
1.3.7 Specification of Abstract Syntax Notation One (ASN.1)	5
1.3.8 Commitment, Concurrency, and Recovery	6
1.3.9 Distributed Transaction Processing	6
1.3.10 Reference Model of Data Management	6
1.3.11 Remote Database Access	6
1.3.11.1 database language	6
1.3.11.2 database language command	6
1.3.11.3 database language statement	6

	Page
1.3.11.4 database server	6
1.3.11.5 data resource	7
1.3.11.6 RDA client	7
1.3.11.7 RDA Control service	7
1.3.11.8 RDA dialogue	7
1.3.11.9 RDA dialogue-state model	7
1.3.11.10 RDA Generic Standard	7
1.3.11.11 RDA operation	7
1.3.11.12 RDA protocol machine	7
1.3.11.13 RDA server	7
1.3.11.14 RDA Service	7
1.3.11.15 RDA Specialization Standard, RDA Specialization	7
1.3.11.16 RDA transaction	7
1.4 Abbreviations	8
1.5 Conventions	9
1.5.1 Service conventions	9
1.5.2 Service parameter description	9
Section 2. Model	
2.1 Concepts	12
2.1.1 Overview of the components	12
2.1.2 Database server concepts	12
2.1.2.1 Organization of data	12
2.1.2.2 RDA transactions	13
2.1.2.3 RDA operations	13
2.1.2.4 Database language commands	14
2.1.3 Communication concepts	14
2.1.3.1 RDA dialogues	14
2.1.3.2 Failure and recovery	15
2.1.4 RDA application-contexts	16
2.1.5 RDA Specialization Standards	16
Section 3. Service	
3.1 Services	18
3.1.1 RDA Dialogue Management services	19

	Page
3.1.1.1 RDA Dialogue Initialization functional unit	19
3.1.1.1.1 R-Initialize service	20
3.1.1.2 RDA Dialogue Termination functional unit	23
3.1.1.2.1 R-Terminate service	23
3.1.2 RDA Transaction Management services	25
3.1.2.1 RDA Transaction Management functional unit	25
3.1.2.1.1 R-Begin Transaction service	25
3.1.2.1.2 R-Commit service	26
3.1.2.1.3 R-Rollback service	28
3.1.3 RDA Control services	29
3.1.3.1 Cancel functional unit	29
3.1.3.1.1 R-Cancel service	29
3.1.3.2 Status functional unit	31
3.1.3.2.1 R-Status service	32
3.1.4 Resource Handling services	34
3.1.4.1 Resource Handling functional unit	35
3.1.4.1.1 R-Open service	35
3.1.4.1.2 R-Close service	38
3.1.5 Database Language services	40
3.1.5.1 Immediate Execution DBL functional unit	41
3.1.5.1.1 R-ExecuteDBL service	41
3.1.5.2 Stored Execution DBL functional unit	43
3.1.5.2.1 R-DefineDBL service	44
3.1.5.2.2 R-InvokeDBL service	46
3.1.5.2.3 R-DropDBL service	49
3.2 Sequencing rules	51
3.2.1 RDA client sequencing rules	51
3.2.2 RDA server sequencing rules	54
Section 4. Protocol	
4.1 Server execution rules	58
4.1.1 RDA dialogue-state model	58
4.1.1.1 RDA operation entity	59
4.1.1.2 RDA dialogue entity	60

		Page
4.1.1.3	Opened data resource entity	61
4.1.1.4	Defined DBL entity	61
4.1.2	General server execution rules	62
4.1.2.1	Generation of the RDA operation entity	62
4.1.2.2	Implementor defined errors	63
4.1.2.3	Beginning of an RDA operation	63
4.1.2.4	Cancellation of an RDA operation	63
4.1.2.5	Execution of an RDA operation	63
4.1.2.6	End of an RDA operation	64
4.1.2.7	Response to an RDA operation	64
4.1.2.8	Failure of the RDA dialogue	65
4.1.3	RDA Dialogue Management services	65
4.1.3.1	RDA Dialogue Initialization functional unit	65
4.1.3.1.1	R-Initialize service	65
4.1.3.2	RDA Dialogue Termination functional unit	66
4.1.3.2.1	R-Terminate service	66
4.1.4	RDA Transaction Management services	66
4.1.4.1	RDA Transaction Management functional unit	66
4.1.4.1.1	R-BeginTransaction service	66
4.1.4.1.2	R-Commit service	67
4.1.4.1.3	R-Rollback service	67
4.1.5	RDA Control services	68
4.1.5.1	Cancel functional unit	68
4.1.5.1.1	R-Cancel service	68
4.1.5.2	Status functional unit	69
4.1.5.2.1	R-Status service	69
4.1.6	Resource Handling services	71
4.1.6.1	Resource Handling functional unit	71
4.1.6.1.1	R-Open service	71
4.1.6.1.2	R-Close service	72
4.1.7	Database Language services	73
4.1.7.1	Immediate Execution DBL functional unit	73
4.1.7.1.1	R-ExecuteDBL service	73
4.1.7.2	Stored Execution DBL functional unit	74

	Page
4.1.7.2.1 R-DefineDBL service	74
4.1.7.2.2 R-InvokeDBL service	75
4.1.7.2.3 R-DropDBL service	76
4.2 RDA protocol machine	77
4.2.1 Functional units	77
4.2.2 Correspondence between RDA service primitives and RDA APDUs	78
4.2.3 Concatenation	78
4.2.4 State tables	79
4.2.4.1 Conventions	79
4.2.4.2 Actions to be taken by the RDAPM	80
4.2.4.3 States	81
4.2.4.4 Incoming events	82
4.2.4.5 Outgoing actions	84
4.2.4.6 Predicates	85
4.2.4.7 RDAPM state tables	86
4.2.4.7.1 RDA client state tables	86
4.2.4.7.2 RDA server state tables	90
4.2.4.7.3 Values of diagnosticInformation for invalidSequence error	94
4.2.5 Protocol procedures	94
4.2.5.1 Initialization of an RDA dialogue	94
4.2.5.2 Termination of an RDA dialogue	94
4.2.5.3 Initiation of an RDA transaction	95
4.2.5.4 Termination of an RDA transaction	95
4.3 Application-protocol-data-units	96
4.4 Conformance	112
4.4.1 Static conformance	112
4.4.2 Dynamic conformance	112
Section 5. Application-contexts	
5.1 RDA Basic application-context	114
5.1.1 Application-context name	114
5.1.2 Purpose and scope	114
5.1.2.1 General description	114
5.1.2.2 RDA dialogue failure	114

		Page
5.1.3	Set of ASEs	114
5.1.4	SACF rules	114
5.1.4.1	Association establishment and release	115
5.1.4.1.1	A-ASSOCIATE	115
5.1.4.1.2	A-RELEASE	115
5.1.4.1.3	A-ABORT	115
5.1.4.1.4	A-P-ABORT	115
5.1.4.2	RDA dialogue initialization and termination	115
5.1.4.2.1	R-Initialize	115
5.1.4.2.2	R-Terminate	116
5.1.4.3	Mapping rules	116
5.1.4.3.1	ACSE APDUs	116
5.1.4.3.2	RDA APDUs	116
5.1.5	State transition diagrams	116
5.1.6	Use of optional features	119
5.1.6.1	A-ASSOCIATE	119
5.1.6.2	A-RELEASE	119
5.1.6.3	A-ABORT	119
5.1.6.4	A-P-ABORT	119
5.1.7	Conformance	119
5.1.7.1	Static conformance	119
5.1.7.2	Dynamic conformance	120
5.2	RDA TP application-context	121
5.2.1	Application-context name	121
5.2.2	Purpose and scope	121
5.2.3	Set of ASEs	121
5.2.4	SACF rules	121
5.2.4.1	Sequencing rules	122
5.2.4.1.1	RDA with TP Dialogue functional unit	122
5.2.4.1.2	RDA with TP Polarized Control functional unit	123
5.2.4.1.3	RDA with TP Shared Control functional unit	123
5.2.4.1.4	RDA with TP Handshake functional unit	123
5.2.4.1.5	RDA with TP Commit and Chained Transactions functional units	123
5.2.4.1.6	RDA with TP Commit and Unchained Transactions functional units	124

	Page
5.2.4.1.7 R-Initialize	124
5.2.4.2 Mapping rules	124
5.2.4.2.1 TP APDUs	124
5.2.4.2.2 RDA APDUs	124
5.2.4.3 Concatenation rules	124
5.2.4.4 Transaction states	125
5.2.4.4.1 RDA with TP Dialogue functional unit	125
5.2.4.4.2 RDA with TP Polarized Control functional unit	125
5.2.4.4.3 RDA with TP Shared Control functional unit	125
5.2.4.4.4 RDA with TP Handshake functional unit	125
5.2.4.4.5 RDA with TP Commit and Chained Transactions functional units	125
5.2.4.4.6 RDA with TP Commit and Unchained Transactions functional units	126
5.2.5 State transition diagrams	126
5.2.6 Use of optional features	131
5.2.6.1 A-ASSOCIATE	131
5.2.7 Conformance	131
5.2.7.1 Static conformance	131
5.2.7.2 Dynamic conformance	131
Section 6. Specializations	
6.1 RDA Specialization Standards	134
6.1.1 General	134
6.1.2 Model	134
6.1.3 Service	134
6.1.4 Protocol	135
6.1.4.1 Server execution rules	135
6.1.4.2 State tables	135
6.1.4.3 Structure of RDA Specialization APDUs	135
6.1.4.4 Conformance	135
6.1.5 Application-contexts	136
Annex A (informative) Relationship to the Application Layer structure	137
A.1 Introduction	137
A.2 RDA as an application-service-element	137
A.3 RDA application-contexts	137

A.4	RDA service-provider	Page 137
Index		141

	Page
Figure 1 — RDA component relationships	12
Figure 2 — Overview of RDA states	15
Figure 3 — Structure of RDA service primitives	19
Figure 4 — Relationship of RDAPM to RDA model	79
Figure 5 — State transition diagram for RDA Basic application-context — RDA client	117
Figure 6 — State transition diagram for RDA Basic application-context — RDA server	118
Figure 7 — State transition diagram for RDA TP application-context — RDA client (Chained Transactions)	127
Figure 8 — State transition diagram for RDA TP application-context — RDA server (Chained Transactions)	128
Figure 9 — State transition diagram for RDA TP application-context — RDA client (Unchained Transactions)	129
Figure 10 — State transition diagram for RDA TP application-context — RDA server (Unchained Transactions)	130
Figure A.1 — The RDA Service viewed in the RDA Basic application-context	138
Figure A.2 — The RDA Service viewed in the RDA TP application-context	139

	Page
Table 1 — RDA functional units and associated RDA services	18
Table 2 — R-Initialize service primitives and their parameters	20
Table 3 — R-Terminate service primitives and their parameters	24
Table 4 — R-BeginTransaction service primitives and their parameters	25
Table 5 — R-Commit service primitives and their parameters	27
Table 6 — R-Rollback service primitives and their parameters	28
Table 7 — R-Cancel service primitives and their parameters	29
Table 8 — R-Status service primitives and their parameters	32
Table 9 — R-Open service primitives and their parameters	35
Table 10 — R-Close service primitives and their parameters	38
Table 11 — R-ExecuteDBL service primitives and their parameters	41
Table 12 — R-DefineDBL service primitives and their parameters	44
Table 13 — R-InvokeDBL service primitives and their parameters	46
Table 14 — R-DropDBL service primitives and their parameters	49
Table 15 — State table for RDA client service primitives	53
Table 16 — State table for RDA server service primitives	55
Table 17 — RDA functional units and associated RDA APDUs	77
Table 18 — States	81
Table 19 — Incoming events: RDA Dialogue Management and RDA Transaction Management services	82
Table 20 — Incoming events: RDA Control, Resource Handling, and Database Language services	83
Table 21 — Outgoing actions: RDA Dialogue Management and RDA Transaction Management services	84
Table 22 — Outgoing actions: RDA Control, Resource Handling, and Database Language services	85
Table 23 — Predicates	85
Table 24 — RDA client state table: RDA Dialogue Management services	86
Table 25 — RDA client state table: RDA Transaction Management services	87
Table 26 — RDA client state table: RDA Control and Resource Handling services	88

	Page
Table 27 — RDA client state table: Database Language services	89
Table 28 — RDA server state table: RDA Dialogue Management services	90
Table 29 — RDA server state table: RDA Transaction Management services	91
Table 30 — RDA server state table: RDA Control and Resource Handling services	92
Table 31 — RDA server state table: Database Language services	93
Table 32 — diagnosticInformation for invalidSequence error	94

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 9579-1 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 21, *Open systems interconnection, data management and open distributed processing*.

ISO/IEC 9579 consists of the following parts, under the general title *Information technology — Open Systems Interconnection — Remote Database Access*:

- *Part 1: Generic Model, Service and Protocol*;
- *Part 2: SQL specialization*.

Annex A of this part of ISO/IEC 9579 is for information only.

Introduction

The Remote Database Access (RDA) standard is one of a set of International Standards produced to facilitate the interworking of computer systems. It is positioned in the Application Layer of the Reference Model of Open Systems Interconnection (OSI) and is related to other OSI standards, as defined in ISO 7498.

The goal of Remote Database Access is to allow, with a minimum of technical agreement outside the interconnection standards, the interconnection of applications and database systems:

- from different manufacturers;
- under different managements;
- of different levels of complexity;
- exploiting different technologies.

An application may itself be a database system and therefore RDA can be used to support multi-database system interworking.

ISO/IEC 9579 defines a service provided to application programs which represents a boundary between the local processing of an application and that part concerned with communications. There is a mapping between the RDA Service elements defined in ISO/IEC 9579 and the services provided by lower layers of the Reference Model of Open Systems Interconnection. This RDA Service, and the lower layer services, may be used to carry database language statements and data between a client application and a database server to enable an application to read and update data in a remote database.

This part of ISO/IEC 9579 is to be used together with an RDA Specialization Standard (specified in some other part of ISO/IEC 9579) to define an RDA application providing interworking with a database management system supporting specific database language.

Section 1. Introduction

1.1 Scope

ISO/IEC 9579 specifies the OSI Remote Database Access (RDA) Service in terms of

- a) the behaviour, as perceived from the OSI environment, of an application-process, called a database server, that provides database storage facilities and database processing services (that is, provides a database management system) to other application-processes; and
- b) the behaviour, as perceived from the OSI environment, of an application-process, called an RDA client, that accesses remote database facilities.

This part of ISO/IEC 9579, called the “RDA Generic Standard”, specifies the general capabilities of an RDA Service. These generic capabilities are intended to be used for interaction with many different database management systems.

Other parts of ISO/IEC 9579, called “RDA Specialization Standards”, pertain to particular database languages, and augment the RDA Generic Standard by specifying how the generic capabilities of RDA are specialized for each of those database languages.

Thus a complete RDA Service is specified, for a given database language, by the combination of two parts of ISO/IEC 9579, one (this part) for the generic capabilities of RDA and a second (another part) for that particular database language.

The resulting specification is an OSI Application Layer standard.

This part of ISO/IEC 9579 describes the generic capabilities of

- a) the model of Remote Database Access (section 2, Model);
- b) the RDA Service that supports interaction between the RDA client and the database server — specifically, its communicating part, called the RDA server (section 3, Service);
- c) the RDA Protocol by which the RDA client and the database server — specifically, the RDA server — communicate with each other (section 4, Protocol); and
- d) the application-contexts of which the RDA Service is a part, including additional rules and the mapping onto underlying services (section 5, Application-contexts).

This part of ISO/IEC 9579 also specifies the rules for defining a specialized RDA Service and constructing an RDA Specialization Standard for it (section 6, Specializations).

ISO/IEC 9579 does not specify individual implementations or products, nor does it constrain the implementation of entities and interfaces within a computer system.

1.2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9579.

At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9579 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO 7498:1984, *Information processing systems — Open Systems Interconnection — Basic Reference Model*.

ISO 7498-3:1989, *Information processing systems — Open Systems Interconnection — Basic Reference Model — Part 3: Naming and Addressing*.

ISO 8327:1987, *Information processing systems — Open Systems Interconnection — Basic connection oriented session protocol specification*.

ISO 8327:1987/Add. 2:—, *Information processing systems — Open Systems Interconnection — Basic connection oriented session protocol specification — Addendum 2: Unlimited user data¹⁾*.

ISO/TR 8509:1987, *Information processing systems — Open Systems Interconnection — Service conventions*.

ISO 8649:1988, *Information processing systems — Open Systems Interconnection — Service definition for the Association Control Service Element*.

ISO 8650:1988, *Information processing systems — Open Systems Interconnection — Protocol specification for the Association Control Service Element*.

ISO 8822:1988, *Information processing systems — Open Systems Interconnection — Connection oriented presentation service definition*.

ISO 8823:1988, *Information processing systems — Open Systems Interconnection — Connection oriented presentation protocol specification*.

ISO 8823:1988/Amd. 2:—, *Information processing systems — Open Systems Interconnection — Connection oriented presentation protocol specification — Amendment 2: Unlimited user data¹⁾*.

ISO/IEC 8824:1990, *Information technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1)*.

ISO/IEC 9545:1989, *Information technology — Open Systems Interconnection — Application Layer structure*.

ISO/IEC 9804:1990, *Information technology — Open Systems Interconnection — Service definition for the Commitment, Concurrency and Recovery service element*.

ISO/IEC 10026-1:1992, *Information technology — Open Systems Interconnection — Distributed Transaction Processing — Part 1: OSI TP Model*.

ISO/IEC 10026-2:1992, *Information technology — Open Systems Interconnection — Distributed Transaction Processing — Part 2: OSI TP Service*.

ISO/IEC 10026-3:1992, *Information technology — Open Systems Interconnection — Distributed Transaction Processing — Part 3: Protocol specification*.

ISO/IEC 10032:—, *Information technology — Open Systems Interconnection — Reference Model of Data Management¹⁾*.

¹⁾ To be published.

1.3 Definitions

For the purposes of this part of ISO/IEC 9579, the following definitions apply.

1.3.1 Basic Reference Model

This part of ISO/IEC 9579 uses the following terms defined in ISO 7498:

- a) application-entity
- b) Application Layer
- c) application-process
- d) application-protocol-data-unit
- e) open system
- f) presentation-connection
- g) transfer syntax

1.3.2 Reference Model — Naming and Addressing

This part of ISO/IEC 9579 uses the following terms defined in ISO 7498-3:

- a) application-entity-invocation-identifier
- b) application-entity-qualifier
- c) application-process-invocation-identifier
- d) application-process-title

1.3.3 Service conventions

This part of ISO/IEC 9579 uses the following terms defined in ISO/TR 8509:

- a) confirm
- b) indication
- c) request
- d) response
- e) service primitive
- f) service-provider
- g) service-user

1.3.4 Application Layer Structure

This part of ISO/IEC 9579 uses the following terms defined in ISO/IEC 9545:

- a) application-association, association
- b) application-context
- c) application-entity-invocation
- d) application-process-invocation
- e) application-service-element
- f) multiple association control function
- g) single association control function
- h) single association object

1.3.5 Connection Oriented Presentation Service Definition

This part of ISO/IEC 9579 uses the following terms defined in ISO 8822:

- a) abstract syntax
- b) presentation data value

1.3.6 Service Definition for the Association Control Service Element

This part of ISO/IEC 9579 uses the following terms defined in ISO 8649:

- a) Default Presentation Context Name
- b) Presentation Context Definition List

1.3.7 Specification of Abstract Syntax Notation One (ASN.1)

This part of ISO/IEC 9579 uses the following terms defined in ISO/IEC 8824:

- a) object descriptor
- b) object identifier
- c) module
- d) tag

1.3.8 Commitment, Concurrency, and Recovery

This part of ISO/IEC 9579 uses the following terms defined in ISO/IEC 9804:

- a) atomicity
- b) consistency
- c) durability
- d) isolation

1.3.9 Distributed Transaction Processing

This part of ISO/IEC 9579 uses the following terms defined in ISO/IEC 10026-1:

- a) control
- b) distributed transaction
- c) local resource
- d) recovery
- e) remote resource
- f) resource
- g) Transaction Processing Service User Invocation

1.3.10 Reference Model of Data Management

This part of ISO/IEC 9579 uses the following term defined in ISO/IEC 10032:

- a) database management system

1.3.11 Remote Database Access

For the purposes of this part of ISO/IEC 9579 the following definitions apply:

1.3.11.1

database language

a definition of the syntax and semantics of operations on a database

1.3.11.2

database language command

a type of RDA operation that models a request for accessing or updating a database. A database language command may have a command handle, always has a database language statement, and may have argument and result specifications

1.3.11.3

database language statement

definition, in a database language, of an operation on a database

1.3.11.4

database server

an application-process that supplies database storage facilities and provides, through OSI communication, database services to other application-processes called RDA clients

1.3.11.5**data resource**

a named collection of data and/or capabilities on the database server known to both the RDA client and the RDA server. The RDA client opens a data resource in order to gain access to its data content or capabilities

NOTE Further capabilities may be defined by RDA Specializations.

1.3.11.6**RDA client**

the RDA service-user that initializes an RDA dialogue and requests database access from a remote database server

1.3.11.7**RDA Control service**

a type of RDA service that allows the RDA client to control outstanding RDA operations. RDA Control services allow an RDA client to a) cancel outstanding RDA operations, and b) query the RDA server for the status of outstanding RDA operations

1.3.11.8**RDA dialogue**

the relationship between an RDA client and an RDA server in which all interactions occur. The RDA dialogue is initialized by the RDA client and has a unique identifier which is assigned by the RDA application-entity-invocation when the RDA dialogue is first initialized

1.3.11.9**RDA dialogue-state model**

a model of the state of an RDA dialogue, defined by a set of entity types and their attributes

1.3.11.10**RDA Generic Standard**

the standard specifying the general capabilities of an RDA Service, which is augmented by an RDA Specialization Standard to form a complete specification of a specialized RDA Service. The RDA Generic Standard is this part of ISO/IEC 9579

1.3.11.11**RDA operation**

a request initiated by the RDA client and transferred to the RDA server for processing

1.3.11.12**RDA protocol machine**

the protocol machine of an RDA application-service-element

1.3.11.13**RDA server**

the RDA service-user within a database server that provides database access to remote RDA clients

1.3.11.14**RDA service**

the set of communication capabilities that are provided to RDA service-users (RDA clients and RDA servers) for the purpose of remote database access

1.3.11.15**RDA Specialization Standard, RDA Specialization**

a standard that augments the RDA Generic Standard by specifying how the generic capabilities of RDA are specialized for a particular database language

1.3.11.16**RDA transaction**

a logically complete unit of processing as determined by the RDA client. RDA transactions are used to guarantee the consistency of remote database processing

1.4 Abbreviations

For the purposes of this part of ISO/IEC 9579, the following abbreviations apply:

ACSE	Association Control Service Element
AE	application-entity
AEI	application-entity-invocation
AP	application-process
APDU	application-protocol-data-unit
API	application-process-invocation
ASE	application-service-element
ASN.1	Abstract Syntax Notation One
CCR	Commitment, Concurrency and Recovery
DBL	database language
MACF	multiple association control function
OSI	Open Systems Interconnection
PM	protocol machine
RDA	Remote Database Access
RDAPM	RDA protocol machine
SACF	single association control function
SAO	single association object
TP	Transaction Processing
TPSUI	Transaction Processing Service User Invocation

1.5 Conventions

1.5.1 Service conventions

ISO/IEC 9579 applies the conventions of ISO/TR 8509, with the following extension:

- RDA response and confirm service primitives are qualified with the term “result” when the result parameter is present and with the term “error” when the error parameter is present. Thus the following terms are used in the RDA Service definition:

- request;
- indication;
- result response;
- result confirm;
- error response; and
- error confirm.

1.5.2 Service parameter description

A tabular format is used to describe the parameters of the RDA service primitives. Each table consists of five columns, one column containing the name of the RDA parameter and a column each for the request (“Req”), indication (“Ind”), response (“Rsp”), and confirm (“Cnf”) primitives.

Each parameter is listed on a separate line.

Some parameters are composed of subparameters. The structure is indicated by indentation of the subparameters beneath the structured parameter. Presence of subparameters is always dependent on presence of the parameter that they appear under (for example, an optional parameter may have subparameters; if the parameter is not supplied, then no subparameters may be supplied).

Some RDA service parameters are named using a “listOf...” convention. Unless otherwise noted, all parameters whose names begin with “ListOf” specify a list of one or more of the item specified after the “listOf”. In descriptions of “list Of” parameters, the word “item” means an instance of the collection composing the “listOf” parameter. This collection consists of all the subparameters shown under the “listof” parameter in the table of service primitives and their parameters.

If “(SPEC)” follows the parameter name, then the meaning and structure of the parameter is defined in an RDA Specialization Standard. Unless the type of usage of such a parameter is “M”, the RDA Specialization Standard may omit the parameter. If such a parameter is included in an RDA Specialization and its parameter usage is “X”, then the RDA Specialization specifies its usage as described later in this clause.

Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the vertical column:

- M — parameter is mandatory for the primitive.
- U — parameter is a user option, and may or may not be provided depending on the dynamic requirements of the RDA client or RDA server.
- C — parameter is conditional, and subject to rules stated in the parameter description.

When a parameter has a “C” usage code in the “Ind” or “Cnf” column, then that parameter only appears on the indication or confirm primitive if the equivalent parameter appeared on the respective request or response primitive.

- S — parameter is a mandatory selection of one from a collection of two or more possible parameters. The parameters that make up this collection are indicated in the table as follows:
- a) each parameter in the collection is specified with the code “S”;
 - b) the name of each parameter in the collection is at the same indentation from the beginning of the parameter column in the table;
 - c) either
 - 1. each parameter is at the leftmost (outer) indentation in the table; or
 - 2. each parameter is part of the same parameter group. A parameter group is a collection of parameters where each group member has a common parent parameter. The parent parameter for any group member is the first parameter above the member that is not indented as far as that member. In the example below, parameterA and parameterB form a parameter group:

parameterX
parameterA
parameterB
parameterY
parameterC

- Informally, wherever only one of a group of parameters is permissible, these all are specified as subordinate to the same “higher level” parameter, marked “S”, and have the same level of indentation.
- X — parameter usage is defined by each RDA Specialization. If such a parameter is included in the RDA Specialization, then it may be made mandatory (“M”), optional (“U”), or conditional (“C”).
- A blank code indicates the parameter is never present.
- The code “(=)” following the usage type indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table. (For instance, an “M(=)” code in the “Ind” column and an “M” in the “Req” column means that the parameter in the indication primitive is semantically equivalent to that in the request primitive.)

Section 2. Model

2.1 Concepts

This clause explains the structure and concepts of the database server, and of the Remote Database Access Service.

2.1.1 Overview of the components

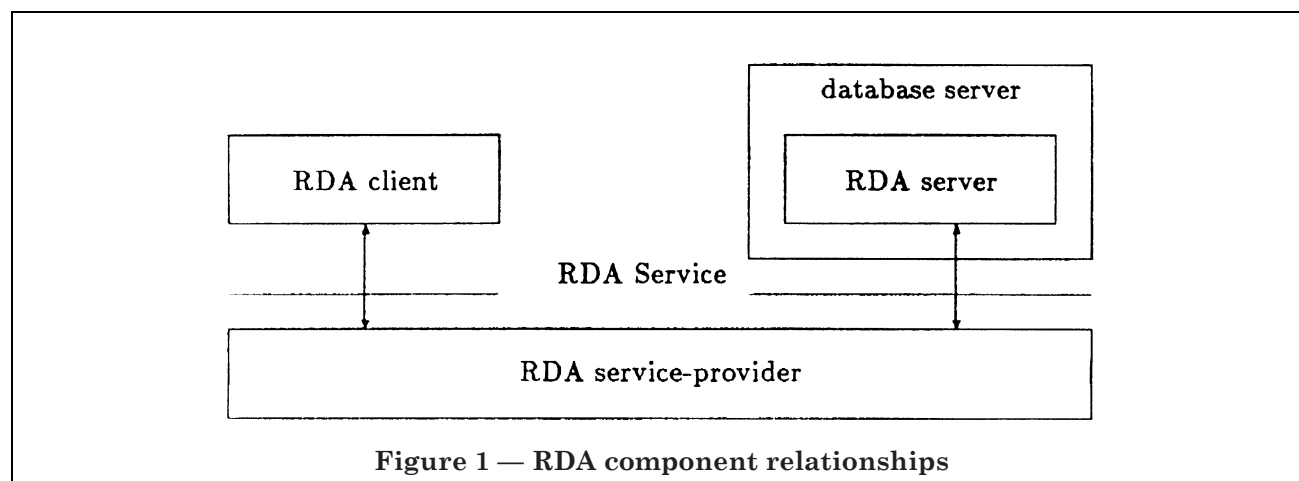
An RDA client is an application-process, within an open system, that requests database services from another application-process (called a database server).

A database server is an application-process, within the same or another open system, that supplies database storage facilities and provides, through OSI communication, database services to RDA clients.

An RDA client and a database server communicate by means of the RDA Service, supported by the RDA service-provider. The part of a database server that uses the RDA service-provider to communicate with an RDA client is called an RDA server.

The RDA client has the ability to initiate RDA service requests, while the RDA server can only issue RDA service responses to reply to such requests.

Figure 1 shows the relationship between these components.



2.1.2 Database server concepts

2.1.2.1 Organization of data

A data resource is a named collection of data and/or capabilities on the database server and known to both the RDA client and the RDA server. The meaning of the data content and capabilities of a data resource depend upon the application of RDA, which is determined by each RDA Specialization Standard.

The RDA client opens a data resource in order to gain access to the data content or capabilities of that data resource through Database Language services. If the RDA client closes a data resource, it no longer has access to that data resource.

Data resources may be nested, with subordinate data resources grouped within their parent data resource. The RDA client is required to open a parent data resource before it can open subordinate data resources. If a parent data resource is closed, all subordinate data resources are also closed.

2.1.2.2 RDA transactions

An RDA transaction is a logically complete unit of processing as determined by the RDA client. Execution during an RDA transaction of a sequence of database access services that change data resources enables the set of changes to be handled as an atomic unit. When the RDA transaction is terminated, either the whole set of changes are applied to the data resource or no changes are applied.

At any time at most one RDA transaction may be processed by the RDA server for a particular RDA dialogue.

NOTE However, within a real open system, an implementation may be processing several RDA transactions concurrently, each of them being related to independent RDA dialogues.

The RDA client requests termination of an RDA transaction by requesting the RDA server either to commit or to roll back the complete set of changes made during that RDA transaction to the data resources involved in that RDA transaction.

The RDA server may initiate rollback of the changes made to the data resources during an RDA transaction. However, the RDA transaction is not terminated until the RDA client explicitly requests its termination.

Changes made to the data content of data resources during an RDA transaction are not made available to other RDA clients until that RDA transaction is terminated at the RDA server.

An RDA transaction has three main states:

- a) RDA transaction not open, when no RDA transaction is in progress;
- b) RDA transaction open, when an RDA transaction is in progress but its termination has not yet been requested; and
- c) RDA transaction terminating, when an RDA transaction is in progress and its termination has been requested.

There are two types of transaction management provided in RDA:

- a) one-phase commitment; and
- b) two-phase commitment.

2.1.2.3 RDA operations

An RDA operation models a request by an RDA client that is transferred to an RDA server for processing. RDA operations enable an RDA client to request any of five types of RDA services:

- a) RDA Dialogue Management services, to start and end RDA dialogues;
- b) RDA Transaction Management services, to start and end RDA transactions;
- c) RDA Control services, to report the status of or to cancel previously issued but not yet completed RDA operations;
- d) Resource Handling services, to enable or disable access by the RDA client to data resources; and
- e) Database Language services, to access and modify data resources.

An RDA client may request RDA operations without waiting for the results of previously requested RDA operations. Thus an RDA server may have several RDA operations outstanding for a particular RDA dialogue; that is, RDA operations for which the RDA server has received an indication service primitive but for which it has not yet issued a response service primitive.

RDA operations are normally executed in the order in which they were received. However, RDA Control service requests may be executed at any time following their receipt, instead of waiting their turn in the queue of outstanding RDA operations.

2.1.2.4 Database language commands

Each RDA Specialization Standard identifies a particular database language (DBL) that is used to express DBL statements. DBL statements are used to access or modify data resources.

In order to execute DBL statements remotely at an RDA server, the RDA client constructs database language commands and transmits them to the RDA server via RDA Service requests. A DBL command may include a command handle, always includes a DBL statement, and may include argument specifications that provide descriptions of the argument values to be used by the DBL statement and result specifications that provide descriptions of the result values returned after processing of the DBL statement. The command handle identifies a particular DBL command.

An RDA client may request execution of a DBL statement in either of two ways:

- a) by requesting execution in a single RDA operation (“execute”). In this case, no command handle is needed; or
- b) by supplying the DBL statement to the RDA server in one RDA operation (“define”) and requesting its execution later in another RDA operation (“invoke”). In this case, a command handle is provided in the define operation and referenced in the invoke operation.

2.1.3 Communication concepts

2.1.3.1 RDA dialogues

An RDA dialogue is a cooperative relationship between an RDA client and an RDA server. The RDA client initializes the RDA dialogue and requests RDA operations that are to be performed by the RDA server. All RDA operations between the RDA client and RDA server occur within the bounds of an RDA dialogue.

An RDA dialogue is uniquely identified within the scope of the OSI environment.

When an RDA dialogue is initialized, the RDA client assigns an RDA dialogue identifier. The RDA dialogue identifier is used for two purposes:

- a) when an RDA dialogue is initialized, to verify the uniqueness of the identifier of the new RDA dialogue; and
- b) when RDA Control services are addressed to another RDA dialogue, to identify that RDA dialogue.

An RDA dialogue has two main states:

- a) inactive, when no RDA dialogue exists between the RDA client and the RDA server; and
- b) active, when an RDA dialogue is established between the RDA client and the RDA server.

Figure 2 expands these main states of an RDA dialogue to show their relationship to the states of an RDA transaction.

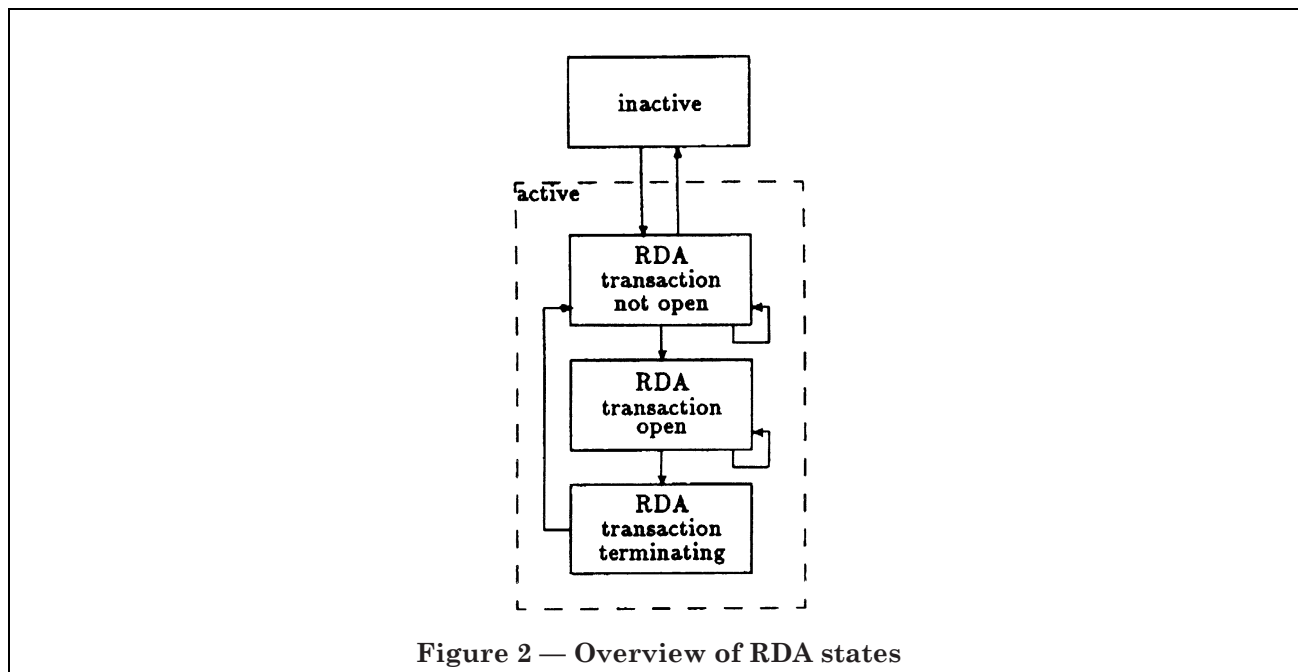


Figure 2 — Overview of RDA states

An RDA dialogue can exist only in the context of an established application-association, and ceases to exist if the association is released (normally or abnormally). Thus the states of an RDA dialogue can be considered as existing within the “associated” state of an application-association. (This relationship has been omitted from Figure 2 for simplicity in presenting the RDA states.)

2.1.3.2 Failure and recovery

An RDA dialogue fails if

- a) the RDA client, the RDA server, or one of their respective RDAPMs is unable to pursue communication on the RDA dialogue; or
- b) a failure in the application-association supporting the RDA dialogue causes an abnormal release of that association.

A failed RDA dialogue cannot be recovered. Recovery actions outside the RDA service-provider may be necessary before the RDA client can initialize a new RDA dialogue. The process of recovery after failure of an RDA dialogue is a local matter and is beyond the scope of this part of ISO/IEC 9579.

However, it is a requirement that all changes made to data resources by any RDA transaction that is not already terminating when RDA dialogue failure occurs be rolled back by the database server during its recovery process. If an RDA transaction is terminating when RDA dialogue failure occurs, then it may be either committed or rolled back.

2.1.4 RDA application-contexts

This part of ISO/IEC 9579 specifies the following generic application-contexts:

- a) RDA Basic application-context: The RDA Basic application-context consists of RDA and ACSE. The type of transaction management is one-phase commitment, and is provided by the RDA Service.

Within the RDA Basic application-context, an RDA dialogue is supported by an association. The RDA client is required to establish that association before the RDA dialogue can be initialized.

- b) RDA TP application-context: The RDA TP application-context consists of RDA, TP, ACSE, and optionally CCR. The type of transaction management is two-phase commitment, and is provided by the TP Service.

Within the RDA TP application-context, an RDA dialogue is supported by a TP dialogue, which in turn is supported by an association. The RDA dialogue is initialized when the TP dialogue is established.

The RDA dialogue is terminated when the TP dialogue is terminated.

2.1.5 RDA Specialization Standards

An RDA Specialization Standard is an elaboration of the RDA Generic Standard to define a specialized RDA Service (an RDA ASE). Each RDA Specialization Standard may define additional constraints on the model, service description, protocol specification, and application-context definition stated in the RDA Generic Standard. These constraints take effect together with all the rules specified by the RDA Generic Standard.

Section 3. Service

3.1 Services

The structure of the RDA Service definition is as follows:

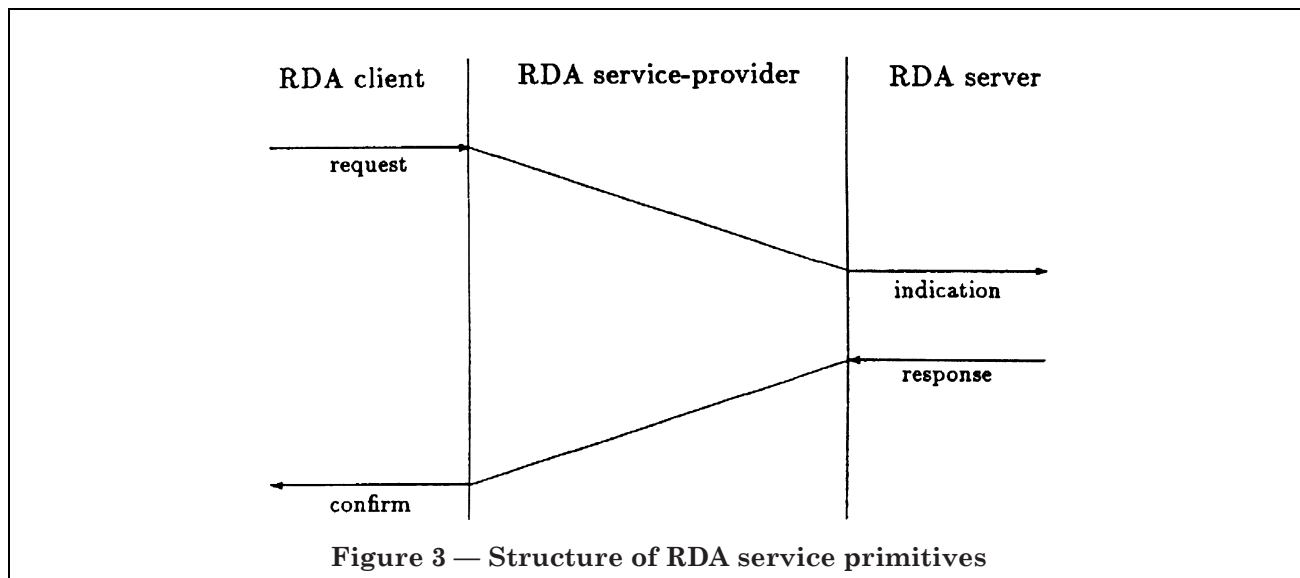
- Classification of the RDA services into groups according to their function in supporting the RDA client's requirements for
 - managing an RDA dialogue;
 - managing RDA transactions;
 - controlling outstanding RDA operations;
 - controlling the availability of data resources;
 - executing database language commands.
- Classification of the RDA services in each group into functional units.
- Description of the RDA services within each functional unit, including the service primitives for each RDA service.
- Definition of each RDA service in the group, consisting of
 - the purpose of the service;
 - a table showing the service parameters, listing the parameters of the request, result response, and error response;
 - descriptions of the request, result response, and error response parameters.

Table 1 lists the RDA functional units and their associated RDA services.

Table 1 — RDA functional units and associated RDA services

Functional unit	Services
RDA Dialogue Initialization	R-Initialize
RDA Dialogue Termination	R-Terminate
RDA Transaction Management	R-BeginTransaction R-Commit R-Rollback
Cancel	R-Cancel
Status	R-Status
Resource Handling	R-Open R-Close
Immediate Execution DBL	R-ExecuteDBL
Stored Execution DBL	R-DefineDBL R-InvokeDBL R-DropDBL

RDA services generally have the structure of service primitives shown in Figure 3.



NOTE 1 All RDA services are confirmed except R-BeginTransaction, which is confirmed only if an error occurs.

Most RDA services may be issued asynchronously. That is, the RDA Service does not generally require that the RDA client wait for a confirm before issuing the next request. Exceptions are R-Initialize, R-Terminate, R-Commit, and R-Rollback.

NOTE 2 The logic of an application may require that the RDA client wait for a confirm, though this is not the concern of the RDA Service.

3.1.1 RDA Dialogue Management services

RDA Dialogue Management services are used to initialize and terminate an RDA dialogue: These services are grouped into two functional units:

- RDA Dialogue Initialization functional unit;
- RDA Dialogue Termination functional unit.

3.1.1.1 RDA Dialogue Initialization functional unit

The RDA Dialogue Initialization functional unit consists of a single service, called R-Initialize, used by an RDA client to initialize an RDA dialogue with an RDA server.

3.1.1.1.1 *R-Initialize service***Purpose**

To initialize an RDA dialogue between the RDA client and the RDA server. This RDA service initializes a new RDA dialogue.

During the initialization of a new RDA dialogue the requesting RDA client may negotiate several parameters.

No further RDA service requests may be issued by an RDA client after issuing this service request until the confirm has been received.

Service parameters

Table 2 lists the R-Initialize service primitives and their parameters.

Table 2 — R-Initialize service primitives and their parameters

Parameter name	Req	Ind	Rsp	Cnf
request	M	M(=)		
operationID	M	M(=)		
dialogueID	M	M(=)		
dialogueID ClientInvocation		M		
aP-title		M		
aE-qualifier		M		
aP-invocationID		M		
aE-invocationID		M		
dialogueIDSuffix	M	M(=)		
identityOfUser	M	M(=)		
userAuthenticationData	U	C(=)		
controlServiceDataRequested	M	M(=)		
functionalUnitsRequested	M	M(=)		
specificInitializeArgument (SPEC)	X	X(=)		
result			S	S(=)
operationID			M	M(=)
controlServiceData			C	C(=)
controlServicesAllowed			M	M(=)
controlAuthenticationData			C	C(=)
functionalUnitsAllowed			M	M(=)
specificInitializeResult (SPEC)			X	X(=)
error			S	S(=)
operationID			M	M(=)
accessControlViolation			S	S(=)
duplicateDialogueID			S	S(=)
invalidSequence				S
diagnosticInformation				U
operationAborted			S	S(=)
errorType			M	M(=)
diagnosticInformation			U	C(=)
userAuthenticationFailure			S	S(=)
specificInitializeError (SPEC)			S	S(=)

Request parameters

operationID: An identifier for the RDA operation which is unique within the RDA dialogue among all currently outstanding RDA operations.

dialogueID: An identifier, which is unique within the OSI environment, used to identify the RDA dialogue being initialized. This parameter consists of the following information:

- **dialogueIDClientInvocation:** The unique identification of the application-entity-invocation of the RDA client that is requesting initialization of the RDA dialogue. This parameter consists of the following information:
 - **aP-title:** The application-process-title of the RDA client.
 - **aE-qualifier:** The application-entity-qualifier of the RDA client.
 - **aP-invocationID:** The application-process-invocation-identifier of the RDA client.
 - **aE-invocationID:** The application-entity-invocation-identifier of the RDA client.

NOTE 1 This information is passed at association establishment time, as specified in section 5, Application-contexts, for each RDA application-context.

- **dialogueIDSuffix:** An identifier which is unique within the scope of the application-entity-invocation (as identified by the dialogueIDClientInvocation parameter) from which the RDA client initializes the RDA dialogue.

identityOfUser: A value that identifies the user of the RDA services. The rules for values of identityOfUser are determined by the RDA server implementation.

NOTE 2 The same identityOfUser value may be used with different RDA clients.

userAuthenticationData: Data used to verify the identity of the user attempting to use RDA services.

controlServiceDataRequested: Indicates whether the RDA client requests that RDA Control services may be used from another RDA dialogue to affect RDA operations on this RDA dialogue. This parameter has one of the following values:

- “false”: Use of RDA Control services from another RDA dialogue is not requested; or
- “true”: Use of RDA Control services from another RDA dialogue is requested.

NOTE 3 Authentication data can be passed to other RDA clients, thereby permitting RDA Control services requested by those other RDA clients to affect RDA operations on this RDA dialogue. The mechanism for passing authentication data between RDA clients is a local matter.

functionalUnitsRequested: The list of functional units requested by the RDA client. This parameter includes one or more of the following values:

- “termination”: the RDA Dialogue Termination functional unit is requested;
- “transaction”: the RDA Transaction Management functional unit is requested;
- “cancel”: the Cancel functional unit is requested;
- “status”: the Status functional unit is requested;
- “resource”: the Resource Handling functional unit is requested;

- “immediate-dbl”: the Immediate Execution DBL functional unit is requested; and
- “stored-dbl”: the Stored Execution DBL functional unit is requested.

NOTE 4 The RDA Dialogue Initialization functional unit is always selected, and therefore is not requested during negotiation.

specificInitializeArgument: This parameter is defined by each RDA Specialization.

Result parameters

operationID: The identifier for the RDA operation to which this is the response.

controlServiceData: Information about the use of RDA Control services from another RDA dialogue to affect RDA operations on this RDA dialogue. This parameter is only returned when controlServiceDataRequested is “true”. It consists of the following information:

- **controlServicesAllowed:** Indicates whether the RDA server has granted permission for this RDA dialogue to be controlled from other RDA dialogues. This parameter has one of the following values:
 - “false”: RDA Control services are not allowed from another RDA dialogue; or
 - “true”: RDA Control services are allowed from another RDA dialogue.

NOTE 5 This permission does not guarantee that an RDA Control service request will succeed at some later time when it is requested from another RDA dialogue.

- **controlAuthenticationData:** The authentication data required when RDA Control services are used from another RDA dialogue to affect RDA operations on this RDA dialogue. This parameter is only returned when controlServicesAllowed is “true”.

functionalUnitsAllowed: Indicates which functional units requested by the RDA client are provided by the RDA server to the RDA client. The list of functional units returned in this parameter is a subset of the list requested by the RDA client via the functionalUnitsRequested parameter.

specificInitializeResult: This parameter is defined by each RDA Specialization.

Error parameters

operationID: The identifier for the RDA operation to which this is the response.

accessControlViolation: The RDA client does not have the required authority to request initialization of the RDA dialogue.

duplicateDialogueID: The dialogueID requested by the RDA client (dialogueIDClientInvocation together with dialogueIDSuffix) duplicates the dialogueID of some other RDA dialogue.

invalidSequence: This RDA service primitive is not allowed in the current state of the RDA server. This parameter also includes the following information:

- **diagnosticInformation:** Supplementary information about why the RDA service primitive is not allowed in the current state of the RDA server. This parameter has one of the following values:
 - “dialogueInitializing”;
 - “dialogueAlreadyActive”; or
 - “dialogueTerminating”.

operationAborted: The RDA operation was aborted. This parameter also includes the following information:

- **errorType:** Indicates whether the error is due to a transient condition of the database server, or is permanent in nature. An errorType of “transient” is used to indicate to the RDA client that the cause of failure is a result of the current dynamic state of the database server, and the same request might succeed at another time. Otherwise the errorType is “permanent”.
- **diagnosticInformation:** Supplementary information about why the RDA operation was aborted. The value of this subparameter is not defined by ISO/IEC 9579.

userAuthenticationFailure: Either the identityOfUser parameter was not recognized or the userAuthenticationData parameter failed to authenticate the user.

specificInitializeError: A specialization-defined error occurred during the processing of an R-Initialize service request. This parameter may also include supplementary information, whose meaning and format is defined by each RDA Specialization.

3.1.1.2 RDA Dialogue Termination functional unit

The RDA Dialogue Termination functional unit consists of a single service, called R-Terminate, used by an RDA client to terminate an existing RDA dialogue with an RDA server.

3.1.1.2.1 R-Terminate service

Purpose

The R-Terminate service enables an RDA client to request termination of an RDA dialogue, by agreement of both the RDA client and the RDA server, without loss of information.

The R-Terminate service is a confirmed service.

An RDA dialogue can not be terminated during an RDA transaction by this service. At the completion of a successful R-Terminate service, all open data resources are closed and all defined DBL commands are deleted.

Service parameters

Table 3 lists the R-Terminate service primitives and their parameters.

Table 3 — R-Terminate service primitives and their parameters

Parameter name	Req	Ind	Rsp	Cnf
request	M	M(=)		
operationID	M	M(=)		
specificTerminateArgument (SPEC)	X	X(=)		
result			S	S(=)
operationID			M	M(=)
specificTerminateResult (SPEC)			X	X(=)

Table 3 — R-Terminate service primitives and their parameters

Parameter name	Req	Ind	Rsp	Cnf
error			S	S(=)
operationID			M	M(=)
duplicateOperationID			S	S(=)
invalidSequence				S
diagnosticInformation				U
operationAborted			S	S(=)
errorType			M	M(=)
diagnosticInformation			U	C(=)
serviceNotNegotiated			S	S(=)
specificTerminateError (SPEC)			S	S(=)

Request parameters

operationID: An identifier for the RDA operation which is unique within the RDA dialogue among all currently outstanding RDA operations.

specificTerminateArgument: This parameter is defined by each RDA Specialization.

Result parameters

operationID: The identifier for the RDA operation to which this is the response.

specificTerminateResult: This parameter is defined by each RDA Specialization.

Error parameters

operationID: The identifier for the RDA operation to which this is the response.

duplicateOperationID: The RDA client has provided an operationID for this RDA operation that is not unique among all outstanding RDA operations within this RDA dialogue.

invalidSequence: This RDA service primitive is not allowed in the current state of the RDA server. This parameter also includes the following information:

- **diagnosticInformation:** Supplementary information about why the RDA service primitive is not allowed in the current state of the RDA server. This parameter has one of the following values:
 - “dialogueNotActive”;
 - “dialogueInitializing”;
 - “transactionOpen”;
 - “transactionTerminating”; or
 - “dialogueTerminating”.

operationAborted: The RDA operation was aborted. This parameter also includes the following information:

- **errorType:** Indicates whether the error is due to a transient condition of the database server, or is permanent in nature. An errorType of “transient” is used to indicate to the RDA client that the cause of failure is a result of the current dynamic state of the database server, and the same request might succeed at another time. Otherwise the errorType is “permanent”.
- **diagnosticInformation:** Supplementary information about why the RDA operation was aborted. The value of this subparameter is not defined by ISO/IEC 9579.

serviceNotNegotiated: An RDA client has issued a request to an RDA server for an RDA service which was not included in the functional units negotiated when the RDA dialogue was initialized.

specificTerminateError: A specialization-defined error occurred during the processing of an R-Terminate service request. This parameter may also include supplementary information, whose meaning and format is defined by each RDA Specialization.

3.1.2 RDA Transaction Management services

RDA Transaction Management services are used to initiate and terminate an RDA transaction. These services are grouped into one functional unit:

- RDA Transaction Management functional unit.

3.1.2.1 RDA Transaction Management functional unit

The RDA Transaction Management functional unit consists of three services, called R-BeginTransaction, R-Commit, and R-Rollback, to respectively initiate, request commitment of, or request rollback of an RDA transaction. The RDA Transaction Management functional unit provides support for one-phase commitment.

3.1.2.1.1 R-BeginTransaction service

Purpose

The R-BeginTransaction service enables an RDA client to initiate an RDA transaction with an RDA server. The R-BeginTransaction service is not confirmed, unless the RDA server encounters an error.

Service parameters

Table 4 lists the R-BeginTransaction service primitives and their parameters.

Table 4 — R-BeginTransaction service primitives and their parameters

Parameter name	Req	Ind	Rsp	Cnf
request	M	M(=)		
operationID	M	M(=)		
error			U	C(=)
operationID			M	M(=)
duplicateOperationID			S	S(=)
invalidSequence				S
diagnosticInformation				U
operationAborted			S	S(=)
errorType			M	M(=)
diagnosticInformation			U	C(=)
serviceNotNegotiated			S	S(=)

Request parameters

operationID: An identifier for the RDA operation which is unique within the RDA dialogue among all currently outstanding RDA operations.

Result parameters

There are no result parameters for this service.

Error parameters

operationID: The identifier for the RDA operation to which this is the response.

duplicateOperationID: The RDA client has provided an operationID for this RDA operation that is not unique among all outstanding RDA operations within this RDA dialogue.

invalidSequence: This RDA service primitive is not allowed in the current state of the RDA server. This parameter also includes the following information:

- **diagnosticInformation:** Supplementary information about why the RDA service primitive is not allowed in the current state of the RDA server. This parameter has one of the following values:
 - “dialogueNotActive”;
 - “dialogueInitializing”;
 - “transactionOpen”;
 - “transactionTerminating”; or
 - “dialogueTerminating”.

operationAborted: The RDA operation was aborted. This parameter also includes the following information:

- **errorType:** Indicates whether the error is due to a transient condition of the database server, or is permanent in nature. An errorType of “transient” is used to indicate to the RDA client that the cause of failure is a result of the current dynamic state of the database server, and the same request might succeed at another time. Otherwise the errorType is “permanent”.
- **diagnosticInformation:** Supplementary information about why the RDA operation was aborted. The value of this subparameter is not defined by ISO/IEC 9579.

serviceNotNegotiated: An RDA client has issued a request to an RDA server for an RDA service which was not included in the functional units negotiated when the RDA dialogue was initialized.

3.1.2.1.2 R-Commit service

Purpose

This RDA service enables an RDA client to request that the current RDA transaction be terminated and that the RDA server attempt to make persistent the set of changes made to data resources during this RDA transaction. If the RDA server is unable to complete this RDA operation, then the RDA transaction is rolled back; that is, none of the changes made to data resources during this RDA transaction are retained.

The R-Commit service is a confirmed service. When successful, the R-Commit response notifies the RDA client whether the RDA transaction was committed or rolled back.

No further RDA operations may be issued by an RDA client after issuing this service request until a confirm indicating commit or rollback has been received.

NOTE The ability of an RDA server to successfully process an R-Commit indication while more than one data resource is open is implementation-dependent.

Service parameters

Table 5 lists the R-Commit service primitives and their parameters.

Table 5 — R-Commit service primitives and their parameters

Parameter name	Req	Ind	Rsp	Cnf
request	M	M(=)		
operationID	M	M(=)		
result			S	S(=)
operationID			M	M(=)
transactionResult			M	M(=)
error			S	S(=)
operationID			M	M(=)
duplicateOperationID			S	S(=)
invalidSequence				S
diagnosticInformation				U

Request parameters

operationID: An identifier for the RDA operation which is unique within the RDA dialogue among all currently outstanding RDA operations.

Result parameters

operationID: The identifier for the RDA operation to which this is the response.

transactionResult: Conveys the resulting disposition of the RDA transaction by the RDA server. This parameter has one of the following values:

- “committed”: all the changes made to data resources since the initiation of the RDA transaction have been secured; or
- “rolledback”: all the changed made to data resources since the initiation of the RDA transaction have been rolled back.

Error parameters

operationID: The identifier for the RDA operation to which this is the response.

duplicateOperationID: The RDA client has provided an operationID for this RDA operation that is not unique among all outstanding RDA operations within this RDA dialogue.

invalidSequence: This RDA service primitive is not allowed in the current state of the RDA server. This parameter also includes the following information:

- **diagnosticInformation:** Supplementary information about why the RDA service primitive is not allowed in the current state of the RDA server. This parameter has one of the following values:
 - “dialogueNotActive”;
 - “dialogueInitializing”;
 - “transactionNotOpen”;
 - “transactionTerminating”; or
 - “dialogueTerminating”.

3.1.2.1.3 R-Rollback service

Purpose

This RDA service enables an RDA client to request that the current RDA transaction be terminated and that none of the changes made to data resources during this RDA transaction be retained.

The R-Rollback service is a confirmed service.

No further RDA operations may be issued by an RDA client after issuing this service request until the confirm has been received.

Service parameters

Table 6 lists the R-Rollback service primitives and their parameters.

Table 6 — R-Rollback service primitives and their parameters

Parameter name	Req	Ind	Rsp	Cnf
request	M	M(=)		
operationID	M	M(=)		
result			S	S(=)
operationID			M	M(=)
error			S	S(=)
operationID			M	M(=)
duplicateOperationID			S	S(=)
invalidSequence				S
diagnosticInformation				U

Request parameters

operationID: An identifier for the RDA operation which is unique within the RDA dialogue among all currently outstanding RDA operations.

Result parameters

operationID: The identifier for the RDA operation to which this is the response.

Error parameters

operationID: The identifier for the RDA operation to which this is the response.

duplicateOperationID: The RDA client has provided an operationID for this RDA operation that is not unique among all outstanding RDA operations within this RDA dialogue.

invalidSequence: This RDA service primitive is not allowed in the current state of the RDA server. This parameter also includes the following information:

- **diagnosticInformation:** Supplementary information about why the RDA service primitive is not allowed in the current state of the RDA server. This parameter has one of the following values:
 - “dialogueNotActive”;
 - “dialogueInitializing”;
 - “transactionNotOpen”;
 - “transactionTerminating”; or
 - “dialogueTerminating”.

3.1.3 RDA Control services

RDA Control services are used to control outstanding RDA operations at the RDA server.

The execution of an RDA operation by the RDA server is not instantaneous and, because of the asynchronous capability of the RDA Service, several RDA operations can be outstanding at one time. Hence, RDA Control services are provided which allow the RDA client to: a) cancel outstanding RDA operations, and b) query the RDA server for the status of particular outstanding RDA operations.

RDA Control services are grouped into two functional units:

- Cancel functional unit;
- Status functional unit.

3.1.3.1 *Cancel functional unit*

The Cancel functional unit consists of a single service, called R-Cancel, used by an RDA client to cancel outstanding RDA operations at an RDA server.

3.1.3.1.1 *R-Cancel service*

Purpose

To request cancellation of all or a selected list of RDA operations. Only Resource Handling and Database Language service requests may be cancelled.

The R-Cancel service response may be returned before the results of previously issued service requests. The R-Cancel result indicates that the R-Cancel request has been processed but does not imply that any RDA operations have been or will be cancelled. Successful cancellation of RDA operations is indicated by subsequent operationCancelled error responses for the cancelled service requests.

Service parameters

Table 7 lists the R-Cancel service primitives and their parameters.

Table 7 — R-Cancel service primitives and their parameters

Parameter name	Req	Ind	Rsp	Cnf
request	M	M(=)		
operationID	M	M(=)		
controlledDialogue	U	C(=)		
dialogueID	M	M(=)		
dialogueIDClientInvocation	U	C(=)		
aP-title	M	M(=)		
aE-qualifier	M	M(=)		
aP-invocationID	M	M(=)		
aE-invocationID	M	M(=)		
dialogueIDSuffix	M	M(=)		
controlAuthenticationData	M	M(=)		
listOfOperationID	U	C(=)		
specificCancelArgument (SPEC)	X	X(=)		

Table 7 — R-Cancel service primitives and their parameters

Parameter name	Req	Ind	Rsp	Cnf
result			S	S(=)
operationID			M	M(=)
specificCancelResult (SPEC)			X	X(=)
error			S	S(=)
operationID			M	M(=)
controlAuthenticationFailure			S	S(=)
controlServicesNotAllowed			S	S(=)
dialogueIDUnknown			S	S(=)
duplicateOperationID			S	S(=)
invalidSequence			S	S
diagnosticInformation				U
operationAborted			S	S(=)
errorType			M	M(=)
diagnosticInformation			U	C(=)
serviceNotNegotiated			S	S(=)
specificCancelError (SPEC)			S	S(=)

Request parameters

operationID: An identifier for the RDA operation which is unique within the RDA dialogue among all currently outstanding RDA operations.

controlledDialogue: A parameter which is used when the R-Cancel service is being used to cancel RDA operations outstanding on a different RDA dialogue. If this parameter is omitted, the R-Cancel service is performed on the current RDA dialogue. This parameter consists of the following information:

- **dialogueID:** An identifier, which is unique within the OSI environment, used to identify the RDA dialogue on which the R-Cancel service is to be performed. This parameter consists of the following information:

- **dialogueIDClientInvocation:** The unique identification of the application-entity-invocation of the RDA client that requested the initialization of the RDA dialogue on which the R-Cancel service is to be performed. If omitted, the same as the current RDA dialogue. This parameter consists of the following information:

- * **aP-title:** The application-process-title of that RDA client.
- * **aE-qualifier:** The application-entity-qualifier of that RDA client.
- * **aP-invocationID:** The application-process-invocation-identifier of that RDA client.
- * **aE-invocationID:** The application-entity-invocation-identifier of that RDA client.

- **dialogueIDSuffix:** The dialogueIDSuffix for the RDA dialogue on which the R-Cancel service is to be performed.

- **controlAuthenticationData:** The authentication data provided by the RDA server in the R-Initialize response for the RDA dialogue on which the R-Cancel service is to be performed.

listOfOperationID: The operationIDs for outstanding RDA operations whose cancellation is requested. If omitted, the cancellation of all outstanding RDA operations is requested. Only Resource Handling and Database Language operations may be cancelled.

specificCancelArgument: The format and meaning of this parameter is defined by each RDA Specialization.

Result parameters

operationID: The identifier for the RDA operation to which this is the response.

specificCancelResult: The format and meaning of this parameter is defined by each RDA Specialization.

Error parameters

operationID: The identifier for the RDA operation to which this is the response.

controlAuthenticationFailure: The controlAuthenticationData parameter does not match the control authentication data of the RDA dialogue referred to by this RDA operation.

controlServicesNotAllowed: The specified controlled dialogue does not permit RDA Control services.

dialogueIDUnknown: The dialogueID parameter value is not equal to the dialogueID of some other RDA dialogue.

duplicateOperationID: The RDA client has provided an operationID for this RDA operation that is not unique among all outstanding RDA operations within this RDA dialogue.

invalidSequence: This RDA service primitive is not allowed in the current state of the RDA server. This parameter also includes the following information:

- **diagnosticInformation:** Supplementary information about why the RDA service primitive is not allowed in the current state of the RDA server. This parameter has one of the following values:
 - “dialogueNotActive”;
 - “dialogueInitializing”;
 - “transactionTerminating”; or
 - “dialogueTerminating”.

operationAborted: The RDA operation was aborted. This parameter also includes the following information:

- **errorType:** Indicates whether the error is due to a transient condition of the database server, or is permanent in nature. An errorType of “transient” is used to indicate to the RDA client that the cause of failure is a result of the current dynamic state of the database server, and the same request might succeed at another time. Otherwise the errorType is “permanent”.
- **diagnosticInformation:** Supplementary information about why the RDA operation was aborted. The value of this subparameter is not defined by ISO/IEC 9579.

serviceNotNegotiated: An RDA client has issued a request to an RDA server for an RDA service which was not included in the functional units negotiated when the RDA dialogue was initialized.

specificCancelError: A specialization-defined error occurred during the processing of an R-Cancel service request. This parameter may also include supplementary information, whose meaning and format is defined by each RDA Specialization.

3.1.3.2 Status functional unit

The Status functional unit consists of a single service, called R-Status, used by an RDA client to determine the status of outstanding RDA operations at an RDA server.

3.1.3.2.1 *R-Status service***Purpose**

To determine the status of one or more outstanding RDA operations. The R-Status service response may be returned before the results of previously issued service requests.

Service parameters

Table 8 lists the R-Status service primitives and their parameters.

Table 8 — R-Status service primitives and their parameters

Parameter name	Req	Ind	Rsp	Cnf
request	M	M(=)		
operationID	M	M(=)		
controlledDialogue	U	C(=)		
dialogueID	M	M(=)		
dialogueIDClientInvocation	U	C(=)		
aP-title	M	M(=)		
aE-qualifier	M	M(=)		
aP-invocationID	M	M(=)		
aE-invocationID	M	M(=)		
dialogueIDSuffix	M	M(=)		
controlAuthenticationData	M	M(=)		
listOfOperationID	U	C(=)		
specificStatusArgument (SPEC)	X	X(=)		
result			S	S(=)
operationID			M	M(=)
listOfStatusInformation			C	C(=)
operationID			M	M(=)
operationStatus			M	M(=)
operationIDUnknown			S	S(=)
awaitingExecution			S	S(=)
executing			S	S(=)
finished			S	S(=)
cancelled			S	S(=)
aborted			S	S(=)
specificOperationStatusResult (SPEC)			X	X(=)
specificStatusResult (SPEC)			X	X(=)
error			S	S(=)
operationID			M	M(=)
controlAuthenticationFailure			S	S(=)
controlServicesNotAllowed			S	S(=)
dialogueIDUnknown			S	S(=)
duplicateOperationID			S	S(=)
invalidSequence				S
diagnosticInformation				U
operationAborted			S	S(=)
errorType			M	M(=)
diagnosticInformation			U	C(=)
serviceNotNegotiated			S	S(=)
specificStatusError (SPEC)			S	S(=)