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

Figures

Tables

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 a specific database language.

Information technology — Open Systems Interconnection — Remote Database Access —

Part 1:
Generic Model, Service and Protocol

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: —[1], *Information processing systems – Open Systems Interconnection – Basic connection oriented session protocol specification – Addendum 2: Unlimited user data.*

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: —[1], *Information processing systems – Open Systems Interconnection – Connection oriented presentation protocol specification – Amendment 2: Unlimited user data.*

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: —[1] *Information technology – Open Systems Interconnection – Reference Model of Data Management.*

[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 "listOf" 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.

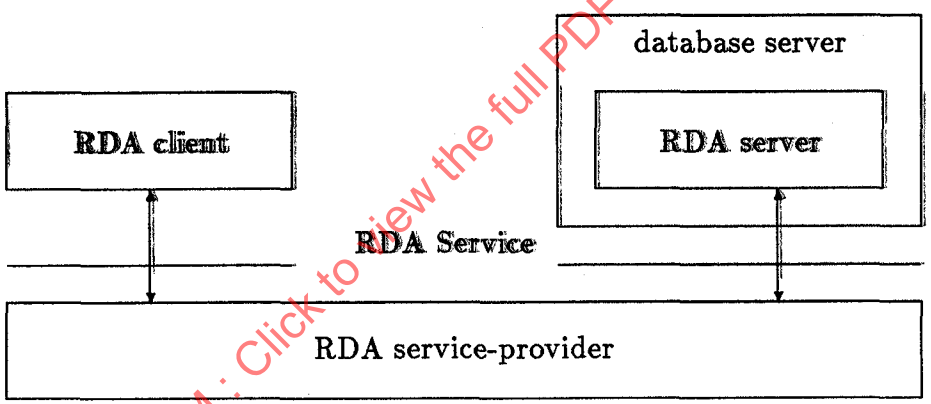

Figure 1 – RDA component relationships

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.

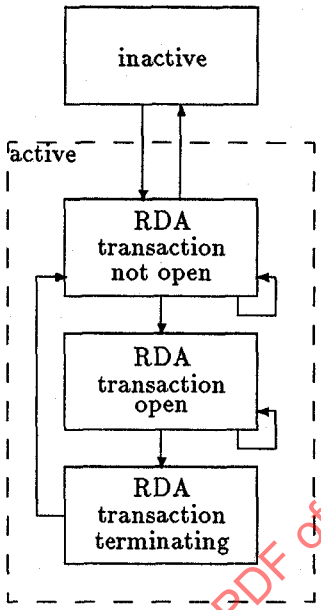

Figure 2 – Overview of RDA states

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.

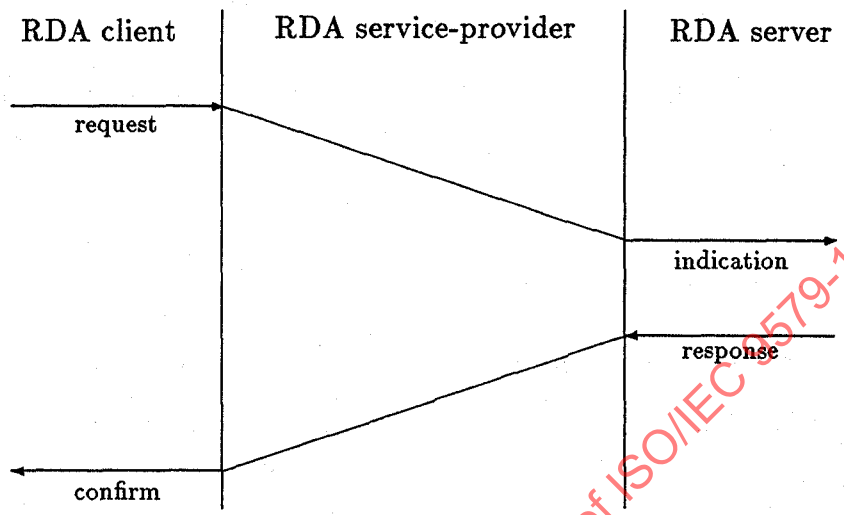

Figure 3 – Structure of RDA service primitives

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(=)		
dialogueIDClientInvocation			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 userAuthentication-Data 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 (concluded)

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 persisent 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 (concluded)

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
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(=)

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-Status service is being used to determine the status of RDA operations outstanding on a different RDA dialogue. If this parameter is omitted, the R-Status 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-Status 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-Status 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-Status 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-Status service is to be performed.

listOfOperationID: The operationIDs for outstanding RDA operations whose status is requested. If omitted, the status of all outstanding RDA operations is requested. Status may be requested for any type of RDA operation.

specificStatusArgument: 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.

listOfStatusInformation: Information about each RDA operation whose status was requested. This parameter is not present if the listOfOperationID request parameter was not specified and there were no outstanding RDA operations at the RDA server for the specified RDA dialogue. This parameter consists of the following information:

- **operationID:** The operationID of an RDA operation whose status is returned.
- **operationStatus:** Indicates the current status of the RDA operation:
  - **operationIDUnknown:** The operationID is not known to the RDA server.
  - **awaitingExecution:** the RDA operation has not yet been executed and is awaiting execution.
  - **executing:** the RDA operation is currently being executed by the database server.
  - **finished:** the RDA operation has completed execution, but the result or error response has not yet been returned.
  - **cancelled:** the RDA operation has been cancelled by the R-Cancel operation, but the error response has not yet been returned.
  - **aborted:** the RDA operation has been aborted, but the error response has not yet been returned.
- **specificOperationStatusResult:** The format and meaning of this parameter is defined by each RDA Specialization.

specificStatusResult: 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.

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

3.1.4 Resource Handling services

Resource Handling services are used to manage the availability of data resources.

The data available at an RDA server may be organized into a set of data resources such that access to the content of a data resource requires that it be opened first.

The resources that are subject to R-Open and R-Close are defined by implementors, subject to constraints that shall be defined in each RDA Specialization Standard.

The result of issuing R-Open and/or R-Close while an RDA transaction is open is subject to constraints that are defined in each RDA Specialization Standard.

Resource Handling services are grouped into one functional unit:

- Resource Handling functional unit.

3.1.4.1 Resource Handling functional unit

The Resource Handling functional unit consists of two services, called R-Open and R-Close, used by the RDA client to control the availability of data resources whose contents are accessed or manipulated or whose capabilities are to be used.

3.1.4.1.1 R-Open service

Purpose

To identify a data resource which can be accessed in succeeding RDA service requests.

The RDA client provides a data resource handle, or identifier, that is not currently being used to identify any other data resource within the RDA dialogue.

A successful R-Open operation makes a data resource available to the RDA client for subsequent use in requests for Database Language services (see 3.1.5, Database Language services), in R-Close requests, or (as a parent data resource) in R-Open requests. The R-Open operation also may restrict the types of DBL statements that may be issued. The handle provided is used to identify the data resource in such subsequent requests.

Service parameters

Table 9 lists the R-Open service primitives and their parameters.

Table 9 – R-Open service primitives and their parameters

Parameter name		Req	Ind	Rsp	Cnf
request		M	M(=)		
operationID		M	M(=)		
dataResourceHandle		M	M(=)		
parentDataResourceHandle		U	C(=)		
dataResourceName		U	C(=)		
specificAccessControlData	(SPEC)	X	X(=)		
specificUsageMode	(SPEC)	X	X(=)		
specificOpenArgument	(SPEC)	X	X(=)		
result				S	S(=)
operationID				M	M(=)
specificOpenResult	(SPEC)			X	X(=)

Table 9 – R-Open service primitives and their parameters (concluded)

Parameter name	Req	Ind	Rsp	Cnf
error			S	S(=)
operationID			M	M(=)
dataResourceAlreadyOpen			S	S(=)
dataResourceHandle			M	M(=)
dataResourceNameNotSpecified			S	S(=)
dataResourceNotAvailable			S	S(=)
errorType			M	M(=)
diagnosticInformation			U	C(=)
dataResourceUnknown			S	S(=)
duplicateDataResourceHandle			S	S(=)
duplicateOperationID			S	S(=)
invalidSequence				S
diagnosticInformation				U
operationAborted			S	S(=)
errorType			M	M(=)
diagnosticInformation			U	C(=)
operationCancelled			S	S(=)
parentDataResourceHandleUnknown			S	S(=)
serviceNotNegotiated			S	S(=)
specificOpenError (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.

dataResourceHandle: The name used by Database Language and Resource Handling services to identify the data resource. The value of this handle shall be unique with respect to other currently valid data resource handles within the scope of the RDA dialogue.

parentDataResourceHandle: The handle of the data resource referenced as the parent data resource.

dataResourceName: Data resources have names by which they are identified for opening purposes. It is possible for an RDA client to open a data resource without supplying a name, provided this is not ambiguous in that user's context.

specificAccessControlData: Data used to verify that the user has authority to open the specified data resource. The structure and meaning of this parameter are defined by each RDA Specialization.

specificUsageMode: The requested access mode(s) to the specified data resource. The access modes that may be specified are defined by each RDA Specialization.

specificOpenArgument: The structure and meaning of this parameter are defined by each RDA Specialization.

Result parameters

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

specificOpenResult: The structure and meaning of this parameter are defined by each RDA Specialization.

Error parameters

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

dataResourceAlreadyOpen: An attempt has been made to open a data resource that is already open on this RDA dialogue. This parameter also includes the following information:

- **dataResourceHandle:** The handle of the data resource already opened.

dataResourceNameNotSpecified: The dataResourceName was not supplied and there are multiple data resources available.

dataResourceNotAvailable: The data resource named in the dataResourceName parameter is known to the RDA server, but is currently not available. The reason may be implementation dependent. This parameter also includes the following information:

- **errorType:** Indicates whether the error is due to a transient condition in 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 data resource is not currently available. The value of this subparameter is not defined by ISO/IEC 9579.

dataResourceUnknown: The data resource named in the dataResourceName parameter is not known to the RDA server.

duplicateDataResourceHandle: The data resource handle provided in the dataResourceHandle parameter is already in use in this 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.

operationCancelled: The RDA operation has been cancelled by the RDA server as the result of an R-Cancel operation.

parentDataResourceHandleUnknown: The parentDataResourceHandle value provided does not correspond to a dataResourceHandle defined in this RDA dialogue.

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.

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

3.1.4.1.2 R-Close service

Purpose

To terminate availability of one or more data resources.

An R-Close operation invalidates one or more data resource handles and thereby makes those data resources unavailable to the RDA client for subsequent use in requests for Database Language services (see 3.1.5, Database Language services) or (as parent data resources) in R-Open requests. It also invalidates any command handles referencing those data resources.

NOTE – Subsequent R-Open operations on the RDA dialogue can re-establish the validity of these data resource handles.

Service parameters

Table 10 lists the R-Close service primitives and their parameters.

Table 10 – R-Close service primitives and their parameters

Parameter name		Req	Ind	Rsp	Cnf
request		M	M(=)		
operationID		M	M(=)		
listOfDataResourceHandle		U	C(=)		
specificCloseArgument	(SPEC)	X	X(=)		
result				S	S(=)
operationID				M	M(=)
listOfCloseExceptions				U	C(=)
dataResourceHandle				M	M(=)
closeException				M	M(=)
dataResourceHandleUnknown				S	S(=)
specificCloseException	(SPEC)			S	S(=)
specificCloseResult	(SPEC)			X	X(=)
error				S	S(=)
operationID				M	M(=)
duplicateOperationID				S	S(=)
invalidSequence					S
diagnosticInformation					U
operationAborted				S	S(=)
errorType				M	M(=)
diagnosticInformation				U	C(=)
operationCancelled				S	S(=)
serviceNotNegotiated				S	S(=)
specificCloseError	(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.

listOfDataResourceHandle: The handles for the set of data resources to be closed. If omitted, all data resources open for this RDA dialogue are to be closed.

If subordinate data resources were opened within a specified data resource using the parentDataResourceHandle parameter of an R-Open request, then those data resources are also closed.

specificCloseArgument: The structure and meaning of this parameter are defined by each RDA Specialization.

Result parameters

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

listOfCloseExceptions: Indicates that exceptions occurred while closing resources. This parameter consists of one or more instances of the following information:

- **dataResourceHandle:** A handle specified in the listOfDataResourceHandle parameter or implied by the absence of a listOfDataResourceHandle parameter whose corresponding data resource was not closed.
- **closeException:** The nature of the close exception. This parameter consists of one of the following values:
  - **dataResourceHandleUnknown:** The dataResourceHandle does not correspond to a data resource handle known to this RDA dialogue; or
  - **specificCloseException:** The format and meaning of this parameter is defined by each RDA Specialization.

specificCloseResult: 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.

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.

operationCancelled: The RDA operation has been cancelled by the RDA server as the result of an R-Cancel operation.

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.

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

3.1.5 Database Language services

Database Language services are used to execute database language (DBL) statements at the database server. The DBL is defined in, or referenced from, each RDA Specialization.

NOTE – The RDA Generic Standard does not constrain the function of DBL statements at the database server; for example, an RDA Specialization may permit DBL statements that operate upon data type information (schemas) as well as application data.

A DBL command defines an operation on data resources that may be performed at the request of the user. A DBL command always includes a DBL statement, may include a command handle, and may include argument specifications and result specifications. The DBL statement and the command handle are supplied by the RDA client.

The argument and result specifications for the DBL statement determine the actual format of the data to be sent and returned when the DBL statement is executed. The specifications for data sent to the RDA server (the "argument specification") are specified by the RDA client. The specifications for the data returned by the RDA server (the "result specification") may be specified either by the RDA client or by the RDA server.

A DBL statement may be provided and executed by an R-ExecuteDBL operation or may be provided by an R-DefineDBL operation and executed by an R-InvokeDBL operation.

The R-DefineDBL operation defines the DBL statement in advance of execution. The DBL statement is given a command handle used to identify it in subsequent R-InvokeDBL operations. A DBL statement defined using the R-DefineDBL operation may be invoked many times with different argument values.

In the R-ExecuteDBL operation the definition and invocation of a DBL statement occur in the same RDA operation, and no handle is allocated.

The ability to intermix invocation (through R-InvokeDBL operations) and execution (through R-ExecuteDBL operations) of DBL statements is subject to constraints that are defined in each RDA Specialization Standard.

A single R-InvokeDBL or R-ExecuteDBL operation may request execution of a DBL statement once or a fixed number of times, depending on either the number of repetitions or the number of multiple arguments specified.

Database Language services are grouped into two functional units:

- Immediate Execution DBL functional unit;
- Stored Execution DBL functional unit.

3.1.5.1 Immediate Execution DBL functional unit

The Immediate Execution DBL functional unit consists of a single service, called R-ExecuteDBL, used by an RDA client to immediately execute a DBL command at an RDA server.

3.1.5.1.1 R-ExecuteDBL service

Purpose

To request the execution of a single DBL statement a fixed number of times and to inform the RDA client of its outcome. The outcome can be complex, in that some executions can succeed and some result in DBL exceptions.

This RDA operation can change the state of data resources.

Service parameters

Table 11 lists the R-ExecuteDBL service primitives and their parameters.

Table 11 – R-ExecuteDBL service primitives and their parameters

Parameter name		Req	Ind	Rsp	Cnf
request		M	M(=)		
operationID		M	M(=)		
dataResourceHandle		U	C(=)		
specificDBLStatement	(SPEC)	M	M(=)		
specificDBLArgumentSpecification	(SPEC)	X	X(=)		
specificDBLResultSpecification	(SPEC)	X	X(=)		
dBLArguments		U	C(=)		
singleArgument		S	S(=)		
repetitionCount		M	M(=)		
specificDBLArgumentValues	(SPEC)	X	X(=)		
multipleArgument		S	S(=)		
listOfSpecificDBLArgumentValues	(SPEC)	M	M(=)		
result				S	S(=)
operationID				M	M(=)
specificTerminationCode	(SPEC)			X	X(=)
specificDBLResultSpecification	(SPEC)			X	X(=)
listOfResultValues				U	C(=)
specificDBLException	(SPEC)			X	X(=)
specificDBLResultValues	(SPEC)			X	X(=)
error				S	S(=)
operationID				M	M(=)
badRepetitionCount				S	S(=)
dataResourceHandleNotSpecified				S	S(=)
dataResourceHandleUnknown				S	S(=)
duplicateOperationID				S	S(=)
invalidSequence					S
diagnosticInformation					U
noDataResourceAvailable				S	S(=)
operationAborted				S	S(=)
errorType				M	M(=)
diagnosticInformation				U	C(=)
operationCancelled				S	S(=)
serviceNotNegotiated				S	S(=)
transactionRolledBack				S	S(=)
specificExecuteDBLError	(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.

dataResourceHandle: Indicates the data resource addressed by the specificDBLStatement. If there is more than one open data resource, this parameter selects the specific open resource. See 3.1.4.1.1, R-Open service.

specificDBLStatement: A DBL statement which, together with a specificDBLArgumentSpecification and a specificDBLResultSpecification, define the database server operation to be executed. The details of this parameter and the rules for determining the corresponding specificDBLArgumentSpecification and specificDBLResultSpecification are defined by each RDA Specialization.

specificDBLArgumentSpecification: The data types of the argument of a DBL statement. The details of this parameter and the rules for determining the corresponding DBL statement are defined by each RDA Specialization.

specificDBLResultSpecification: The data types of the result of a DBL statement, as preferred by the RDA client. The details of this parameter and the rules for determining the corresponding DBL statement are defined by each RDA Specialization.

dBLArguments: Indicates whether or not the specificDBLStatement is executed more than once.

If **singleArgument** is specified, then the specificDBLStatement shall be executed **repetitionCount** times, using specificDBLArgumentValues, if specified, repeatedly for each execution. If **multipleArgument** is specified and listOfSpecificDBLArgumentValues contains **n** items, then the specificDBLStatement shall be executed **n** times, using each item once. If neither singleArgument nor multipleArgument is specified, then the specificDBLStatement shall be executed exactly once.

specificDBLArgumentValues: The actual parameters for a sequence of single DBL statement requests. The structure and meaning of the values are determined by a specificDBLArgumentSpecification parameter that is associated with this DBL statement, according to rules defined in each RDA Specialization.

Result parameters

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

specificTerminationCode: The reason for terminating the execution of a sequence of statements. The meaning of the termination code values is defined in each RDA Specialization.

specificDBLResultSpecification: The data types of the result of a DBL statement, as returned by the RDA server. The details of this parameter and the rules for determining the corresponding DBL statement are defined by each RDA Specialization.

listOfResultValues: The result of each execution of the DBL statement. The number of items in this parameter shall equal the repetitionCount (if singleArgument was specified) or the number of items in the listOfSpecificDBLArgumentValues (if multipleArgument was specified) or one (if neither singleArgument or multipleArgument was specified). This parameter consists of the following information for each execution:

- **specificDBLException:** Indicates normal or abnormal termination of the DBL statement. The details of this parameter are defined by each RDA Specialization.
- **specificDBLResultValues:** The actual values returned from a single DBL statement execution. The structure and meaning of the values are determined by a specificDBLResultSpecification parameter that is associated with this DBL statement, according to rules defined in each RDA Specialization.

Error parameters

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

badRepetitionCount: The repetitionCount provided was less than one.

dataResourceHandleNotSpecified: The RDA client has not provided a dataResourceHandle parameter when there are multiple open data resources in this RDA dialogue.

dataResourceHandleUnknown: The dataResourceHandle value provided does not correspond to a data resource handle known to this 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".

noDataResourceAvailable: A database language operation which does not specify a data resource handle was issued when there are no open data resources for this RDA dialogue.

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.

operationCancelled: The RDA operation has been cancelled by the RDA server as the result of an R-Cancel operation.

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.

transactionRolledBack: The current RDA transaction is rolled back.

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

3.1.5.2 Stored Execution DBL functional unit

The Stored Execution DBL functional unit consists of three services, called R-DefineDBL, R-InvokeDBL, and R-DropDBL, used by an RDA client to (respectively) define and store a DBL command at an RDA server for later execution, execute the stored DBL command, and drop a stored DBL command when it is no longer required.

3.1.5.2.1 R-DefineDBL service

Purpose

For the RDA client to define and the database server to store a specific DBL command.

A defined DBL command has a command handle supplied by the RDA client. The command handle is valid until either it is dropped by R-DropDBL, or the referenced data resource is closed by R-Close, or the RDA dialogue is terminated.

Service parameters

Table 12 lists the R-DefineDBL service primitives and their parameters.

Table 12 – R-DefineDBL service primitives and their parameters

Parameter name		Req	Ind	Rsp	Cnf
request		M	M(=)		
operationID		M	M(=)		
commandHandle		M	M(=)		
dataResourceHandle		U	C(=)		
specificDBLStatement	(SPEC)	M	M(=)		
specificDBLArgumentSpecification	(SPEC)	X	X(=)		
specificDBLResultSpecification	(SPEC)	X	X(=)		
result				S	S(=)
operationID				M	M(=)
specificDBLResultSpecification	(SPEC)			X	X(=)
specificDBLException	(SPEC)			X	X(=)
error				S	S(=)
operationID				M	M(=)
dataResourceHandleNotSpecified				S	S(=)
dataResourceHandleUnknown				S	S(=)
duplicateCommandHandle				S	S(=)
duplicateOperationID				S	S(=)
invalidSequence					S
diagnosticInformation					U
noDataResourceAvailable				S	S(=)
operationAborted				S	S(=)
errorType				M	M(=)
diagnosticInformation				U	C(=)
operationCancelled				S	S(=)
serviceNotNegotiated				S	S(=)
specificDefineDBLError	(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.

commandHandle: The identifier of the defined DBL command that is available for use by subsequent R-InvokeDBL or R-DropDBL operations. Within a single RDA dialogue the value of this handle shall be unique with respect to other currently valid command handles.

dataResourceHandle: The data resource addressed by the specificDBLStatement. If there is more than one open data resource, this parameter selects the specific open resource. See 3.1.4.1.1, R-Open service.

specificDBLStatement: A DBL statement which, together with a specificDBLArgumentSpecification and a specificDBLResultSpecification, define the DBL command to be stored. The details of this parameter are defined by each RDA Specialization.

specificDBLArgumentSpecification: The data types of the argument of the DBL statement. The details of this parameter and the rules for determining the corresponding DBL statement are defined by each RDA Specialization.

specificDBLResultSpecification: The data types of the result of a DBL statement, as preferred by the RDA client. The details of this parameter and the rules for determining the corresponding DBL statement are defined by each RDA Specialization.

Result parameters

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

specificDBLResultSpecification: The data types of the result of a DBL statement, as returned by the RDA server. The details of this parameter and the rules for determining the corresponding DBL statement are defined by each RDA Specialization.

specificDBLException: Used to indicate the outcome of the definition and storage of the DBL statement. The details of this parameter are defined by each RDA Specialization.

NOTE – The commandHandle is still valid even if a specificDBLException is returned.

Error parameters

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

dataResourceHandleNotSpecified: The RDA client has not provided a dataResourceHandle parameter when there are multiple open data resources in this RDA dialogue.

dataResourceHandleUnknown: The dataResourceHandle value provided does not correspond to a data resource handle known to this RDA dialogue.

duplicateCommandHandle: The command handle provided by the RDA client is already in use in this 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".

noDataResourceAvailable: A database language operation which does not specify a data resource handle was issued when there are no open data resources for this RDA dialogue.

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.

operationCancelled: The RDA operation has been cancelled by the RDA server as the result of an R-Cancel operation.

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.

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

3.1.5.2.2 R-InvokeDBL service

Purpose

To execute an already stored DBL command a fixed number of times and to inform the RDA client of its outcome.

Service parameters

Table 13 lists the R-InvokeDBL service primitives and their parameters.

Table 13 – R-InvokeDBL service primitives and their parameters

Parameter name		Req	Ind	Rsp	Cnf
request		M	M(=)		
operationID		M	M(=)		
commandHandle		M	M(=)		
dBLArguments		U	C(=)		
singleArgument		S	S(=)		
repetitionCount		M	M(=)		
specificDBLArgumentValues	(SPEC)	X	X(=)		
multipleArgument		S	S(=)		
listOfSpecificDBLArgumentValues	(SPEC)	M	M(=)		
result				S	S(=)
operationID				M	M(=)
specificDBLResultSpecification	(SPEC)			X	X(=)
specificTerminationCode	(SPEC)			X	X(=)
listOfResultValues				U	C(=)
specificDBLException	(SPEC)			X	X(=)
specificDBLResultValues	(SPEC)			X	X(=)

Table 13 – R-InvokeDBL service primitives and their parameters (concluded)

Parameter name	Req	Ind	Rsp	Cnf
error			S	S(=)
operationID			M	M(=)
badRepetitionCount			S	S(=)
commandHandleUnknown			S	S(=)
duplicateOperationID			S	S(=)
invalidSequence				S
diagnosticInformation				U
operationAborted			S	S(=)
errorType			M	M(=)
diagnosticInformation			U	C(=)
operationCancelled			S	S(=)
serviceNotNegotiated			S	S(=)
transactionRolledBack			S	S(=)
specificInvokeDBLError (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.

commandHandle: The handle of the defined DBL command that is being invoked.

dBLArguments: Indicates whether or not the specificDBLStatement is executed more than once.

If **singleArgument** is specified, then the specificDBLStatement shall be executed **repetitionCount** times, using specificDBLArgumentValues, if specified, repeatedly for each execution. If **multipleArgument** is specified and listOfSpecificDBLArgumentValues contains **n** items, then the specificDBLStatement shall be executed **n** times, using each item once. If neither singleArgument nor multipleArgument is specified, then the specificDBLStatement shall be executed exactly once.

specificDBLArgumentValues: The actual parameters for a sequence of single DBL statement requests. The structure and meaning of the values are determined by a specificDBLArgumentSpecification parameter that is associated with this DBL statement, according to rules defined in each RDA Specialization.

Result parameters

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

specificDBLResultSpecification: The data types of the result of a DBL statement, as returned by the RDA server. The details of this parameter and the rules for determining the corresponding DBL statement are defined by each RDA Specialization.

specificTerminationCode: The reason for terminating the execution of a sequence of statements. The meaning of the termination code values is defined by each RDA Specialization.

listOfResultValues: The result of each execution of the referenced DBL command. The number of items in this parameter shall equal the repetitionCount (if singleArgument was specified) or the number of items in the listOfSpecificDBLArgumentValues (if multipleArgument was specified) or one (if neither singleArgument or multipleArgument was specified). This parameter consists of the following information for each execution:

- **specificDBLException:** Indicates normal or abnormal termination of the DBL statement. The details of this parameter are defined by each RDA Specialization.

- **specificDBLResultValues:** The actual values returned from a single DBL statement execution. The structure and meaning of the values are determined by a specificDBLResultSpecification parameter that is associated with this DBL statement, according to rules defined in each RDA Specialization.

Error parameters

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

badRepetitionCount: The repetitionCount provided was less than one.

commandHandleUnknown: The commandHandle provided does not correspond to a commandHandle currently defined in this 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.

operationCancelled: The RDA operation has been cancelled by the RDA server as the result of an R-Cancel operation.

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.

transactionRolledBack: The current RDA transaction is rolled back.

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

3.1.5.2.3 R-DropDBL service

Purpose

To remove one or more defined DBL commands from the set of defined DBL commands known to the RDA server.

Service parameters

Table 14 lists the R-DropDBL service primitives and their parameters.

Table 14 – R-DropDBL service primitives and their parameters

Parameter name		Req	Ind	Rsp	Cnf
request		M	M(=)		
operationID		M	M(=)		
listOfCommandHandle		U	C(=)		
specificDropDBLArgument	(SPEC)	X	X(=)		
result				S	S(=)
operationID				M	M(=)
listOfDropDBLExceptions				U	C(=)
commandHandle				M	M(=)
dropDBLException				M	M(=)
commandHandleUnknown				S	S(=)
specificDropDBLException	(SPEC)			S	S(=)
specificDropDBLResult	(SPEC)			X	X(=)
error				S	S(=)
operationID				M	M(=)
duplicateOperationID				S	S(=)
invalidSequence					S
diagnosticInformation					U
operationAborted				S	S(=)
errorType				M	M(=)
diagnosticInformation				U	C(=)
operationCancelled				S	S(=)
serviceNotNegotiated				S	S(=)
specificDropDBLError	(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.

listOfCommandHandle: The command handles for DBL commands which are to be dropped. If omitted, all currently defined DBL commands are to be dropped.

specificDropDBLArgument: 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.

listOfDropDBLExceptions: Indicates that exceptions occurred while dropping DBL commands. This parameter consists of one or more instances of the following information:

- **commandHandle:** A handle specified in the listOfCommandHandle parameter or implied by the absence of a listOfCommandHandle parameter whose corresponding DBL command was not dropped.
- **dropDBLException:** The nature of the drop DBL exception. This parameter consists of one of the following values:
  - **commandHandleUnknown:** The commandHandle value provided does not correspond to a DBL command currently defined in this RDA dialogue; or
  - **specificDropDBLException:** The format and meaning of this parameter is defined by each RDA Specialization.

specificDropDBLResult: 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.

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.

operationCancelled: The RDA operation has been cancelled by the RDA server as the result of an R-Cancel operation.

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.

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

3.2 Sequencing rules

This clause defines the allowed sequences of RDA service primitives for an RDA dialogue by means of two state tables. One state table shows sequences for RDA client service primitives (requests and confirms) and the other state table shows sequences for RDA server service primitives (indications and responses).

The intersection of a service primitive event (row) and a state (column) forms a cell. If the cell is not blank, then the service event is permitted in that state, and the entry in the cell specifies the state entered when the event occurs. An entry after a slash (/), shown on a second line for a cell, specifies the state entered when an error occurs (an error response primitive is issued by the RDA server or an error confirm primitive is received by the RDA client, instead of a result response or result confirm respectively).

If the cell is blank, then the service event is not permitted in that state; it is a local matter as to how this error should be resolved.

3.2.1 RDA client sequencing rules

The RDA Service is asynchronous; that is, the RDA client need not wait for a confirm related to one service request before it issues another service request.

However, the RDA client is expected to wait in the following situations:

a) If an RDA client issues an R-Initialize, R-Terminate, R-Commit, or R-Rollback request, then the RDA client should wait for the confirm primitive for that request before issuing any other request.

NOTE 1 – If the RDA service-provider detects that the RDA client has not waited, then it issues an error confirm primitive with the error invalidSequence.

b) If an RDA client receives a transactionRolledBack error confirm to an R-ExecuteDBL or R-InvokeDBL request and the RDA client has not requested termination of the RDA transaction via an R-Commit or R-Rollback request, then the RDA client is required to issue such an R-Commit or R-Rollback request in order to terminate the RDA transaction.

NOTE 2 – The RDA client does not receive confirms thereafter to any RDA services requested before the request to terminate the RDA transaction, since either such service requests are discarded by the RDA service-provider or the corresponding indications are discarded by the RDA server.

The confirms that correspond to service requests are normally received in the same sequence as the requests. The exceptions are

a) R-Status and R-Cancel confirms occur prior to the confirms for any requests on the same RDA dialogue that they report status of or cancel, respectively.

b) InvalidSequence error confirms may occur prior to confirms for earlier requests.

c) If an RDA client receives an error confirm to an R-BeginTransaction request or a transactionRolledBack error confirm to an R-ExecuteDBL or R-InvokeDBL request, then the RDA client does not receive confirms to RDA service requests that were issued subsequent to that request.

NOTE 3 – Either such service requests are discarded by the RDA service-provider or the corresponding indications are discarded by the RDA server.

Table 15 defines the allowed sequences of RDA client service primitives. In this table, the following types of events are shown:

- An RDA request primitive issued by the RDA client to the RDA ASE
- An RDA confirm primitive issued by the RDA ASE to the RDA client

The states have the following meanings:

I – There is no RDA dialogue between an RDA client and an RDA server.

CB – The R-Initialize confirm is pending.

CA – An RDA dialogue is active and there is no RDA transaction in progress.

CT – An RDA dialogue is active and there is an RDA transaction in progress.

CN – A transactionRolledBack error confirm has been received, but no R-Rollback or R-Commit request has been issued.

CC – An RDA dialogue is active, and an R-Commit request has been issued but the confirm is pending.

CR – An RDA dialogue is active, and an R-Rollback request has been issued but the confirm is pending.

CE – An R-Terminate request has been issued but the confirm is pending.

Table 15 – State table for RDA client service primitives

Service Primitive Event	State							
	I	CB	CA	CT	CN	CC	CR	CE
R-Initialize req	CB							
R-Initialize cnf		CA /I						
R-Terminate req			CE					
R-Terminate cnf								I /CA
R-BeginTrans req			CT					
R-BeginTrans cnf				/CA		/CA	/CA	
R-Commit req				CC	CC			
R-Commit cnf						CA /CT or CN		
R-Rollback req				CR	CR			
R-Rollback cnf							CA /CT or CN	
R-Cancel req			CA	CT				
R-Cancel cnf			CA	CT		CC	CR	CE
R-Status req			CA	CT				
R-Status cnf			CA	CT		CC	CR	CE
R-Open req			CA	CT				
R-Open cnf			CA	CT		CC	CR	CE
R-Close req			CA	CT				
R-Close cnf			CA	CT		CC	CR	CE
R-ExecuteDBL req			CA	CT				
R-ExecuteDBL cnf			CA	CT /CT or CN		CC	CR	CE
R-DefineDBL req			CA	CT				
R-DefineDBL cnf			CA	CT		CC	CR	CE
R-InvokeDBL req			CA	CT				
R-InvokeDBL cnf			CA	CT /CT or CN		CC	CR	CE
R-DropDBL req			CA	CT				
R-DropDBL cnf			CA	CT		CC	CR	CE

3.2.2 RDA server sequencing rules

The RDA Service is asynchronous; that is, the RDA server may receive another indication before it issues the response to a previous indication.

However, the order in which the RDA server shall issue responses is constrained by the following rules:

a) Following either an error response to an R-BeginTransaction indication or a transactionRolledBack error response (to an R-ExecuteDBL or R-InvokeDBL indication), the RDA server shall discard any previously received indications.

b) Other than as stated above, the RDA server shall issue responses to R-Cancel and R-Status indications earlier than responses to other indications.

c) The RDA server shall issue responses to all indications other than R-Cancel and R-Status in the order in which the indications were received.

Table 16 defines the allowed sequences of RDA server service primitives. In this table, the following types of events are shown:

- An RDA indication primitive issued by the RDA ASE to the RDA server
- An RDA response primitive issued by the RDA server to the RDA ASE

The states have the following meanings:

I – There is no RDA dialogue between an RDA client and an RDA server.

SB – The R-Initialize response is pending.

SA – An RDA dialogue is active and there is no RDA transaction in progress.

ST – An RDA dialogue is active and there is an RDA transaction in progress.

SN – A transactionRolledBack error response has been sent, but no R-Rollback or R-Commit indication has been received.

SC – An RDA dialogue is active, and an R-Commit indication has been received but the response is pending.

SR – An RDA dialogue is active, and an R-Rollback indication has been received but the response is pending.

SE – An R-Terminate indication has been received but the response is pending.

Table 16 – State table for RDA server service primitives

Service Primitive Event	State							
	I	SB	SA	ST	SN	SC	SR	SE
R-Initialize ind	SB							
R-Initialize rsp		SA /I						
R-Terminate ind			SE					
R-Terminate rsp								I /SA
R-BeginTrans ind			ST					
R-BeginTrans rsp				/SA		/SA	/SA	
R-Commit ind				SC	SC			
R-Commit rsp						SA /ST or SN		
R-Rollback ind				SR	SR			
R-Rollback rsp							SA /ST or SN	
R-Cancel ind			SA	ST				
R-Cancel rsp			SA	ST		SC	SR	SE
R-Status ind			SA	ST				
R-Status rsp			SA	ST		SC	SR	SE
R-Open ind			SA	ST				
R-Open rsp			SA	ST		SC	SR	SE
R-Close ind			SA	ST				
R-Close rsp			SA	ST		SC	SR	SE
R-ExecuteDBL ind			SA	ST				
R-ExecuteDBL rsp			SA	ST /ST or SN		SC	SR	SE
R-DefineDBL ind			SA	ST				
R-DefineDBL rsp			SA	ST		SC	SR	SE
R-InvokeDBL ind			SA	ST				
R-InvokeDBL rsp			SA	ST /ST or SN		SC	SR	SE
R-DropDBL ind			SA	ST				
R-DropDBL rsp			SA	ST		SC	SR	SE

Section 4: Protocol

This section specifies the behaviour of both the RDA service-provider and the RDA server. It consists of the following clauses:

- Clause 4.1, Server execution rules, which specifies the behaviour of the RDA server. It specifies the constraints on an RDA server issuing response primitives (result responses or error responses) in response to indication primitives that it receives from the RDA service-provider, including the permitted values of the parameters of those responses.
- Clause 4.2, RDA protocol machine. This clause specifies the relationship of the RDA Service to the RDA Protocol (the functional units and the names of the APDUs and their fields). It also specifies the behaviour of the RDA protocol machine (the states, the events that may occur, the actions that are performed, and the state transitions).
- Clause 4.3, Application-protocol-data-units, which specifies the structure and content of the RDA application-protocol-data-units, using Abstract Syntax Notation One (ASN.1).
- Clause 4.4, Conformance, which states the conformance requirements that are common to all RDA application-contexts. (Additional conformance requirements specific to the various RDA application-contexts are stated in section 5, Application-contexts.)

4.1 Server execution rules

This clause defines constraints on an RDA server responding to requests from an RDA client. Requests are communicated to the RDA server as indication events. These events, together with result response and error response primitives, define the RDA Service at the RDA server.

Within the RDA server, the requests received from the RDA client are generally modelled as RDA operation entities that progress through the RDA server until they are issued as result or error response primitives by the RDA server. However, requests received from the RDA client that violate sequencing rules are handled by the RDA service-provider and no indications are issued to the RDA server.

The semantics of RDA operation entities and their processing by the database server are defined by a model of the state of the RDA server and by rules governing the progression of RDA operation entities through the RDA server.

The state of the RDA server has two main components:

- The RDA dialogue state models the interactions with a single RDA client over one RDA dialogue.
- The database state models persistent (stored) data that is shared among all the database users. This state is changed by DBL statements as defined by RDA Specializations.

The server execution rules describe the permitted effects of execution of RDA operations by the RDA server. They consist of

- entity manipulation rules, which describe the changes to the RDA dialogue state resulting from execution of an RDA operation entity,
- result rules, which define the conditions that apply when a result response is generated, and
- error rules, which define the conditions that apply when an error response is generated.

An RDA Specialization is permitted to specify additional constraints on the state resulting from successful execution, and on the result which is generated. It is also permitted to specify additional error parameters and error rules.

4.1.1 RDA dialogue-state model

The RDA dialogue-state model is a model used to represent the state of an RDA dialogue. The RDA dialogue state is updated by the RDA server during the execution of RDA operations. Changes to the RDA dialogue state within the RDA server are defined for each RDA operation whose execution changes the RDA dialogue state.

RDA dialogue state information is known to both the RDA client and the RDA server. However, because there are potential communications and processing delays, the RDA client and RDA server may have different information about the state of an RDA dialogue at any instant of time. The RDA client may update its view of the RDA dialogue state from information that flows as result and error responses from the RDA server to the RDA client. However, strict formalization of the RDA dialogue state as known by the RDA client is not defined by this part of ISO/IEC 9579.

The purpose of the model is solely to assist in the specification of the server execution rules. There is no constraint on the way in which implementors represent the information.

The model consists of the following entities:

- RDA operation;
- RDA dialogue;
- opened data resource; and
- defined DBL.

Each of the entity types has a set of associated attributes whose meaning is defined in this clause.

An RDA Specialization Standard shall reference this model, its entities, and their associated generic attributes when defining additional server execution rules. In addition, an RDA Specialization Standard may augment this model by the addition of specific attributes for any of the generic entities, or by the addition of new entities and attributes appropriate to that RDA Specialization.

4.1.1.1 RDA operation entity

An RDA operation entity is created for each RDA service indication received by the RDA server. An RDA operation entity is deleted when a result response or error response primitive is issued for the RDA operation corresponding to that entity.

The RDA operation entity has the following attributes:

a) **dialogueID:** An identifier, unique within the OSI environment, used to identify the RDA dialogue.

b) **operationID:** An identifier, unique within the scope of the RDA dialogue, that identifies the RDA operation.

c) **status:** The current status of the RDA operation:

- **awaitingExecution:** the RDA operation has not yet been executed and is awaiting execution.
- **executing:** the RDA operation is currently being executed by the database server.
- **finished:** the RDA operation has completed execution, but the result response or error response has not yet been returned.
- **cancelled:** the RDA operation has been cancelled by an R-Cancel operation, but the error response has not yet been returned.
- **aborted:** the RDA operation has been aborted, but the error response has not yet been returned.

d) **cancelRequestReceived:** A boolean attribute which is "true" if an R-Cancel service request has been received for this RDA operation.

e) **operationType:** The type of the RDA service indication primitive that generated the RDA operation entity:

- **R-Initialize**
- **R-Terminate**
- **R-BeginTransaction**
- **R-Commit**
- **R-Rollback**
- **R-Cancel**
- **R-Status**

- **R-Open**
- **R-Close**
- **R-ExecuteDBL**
- **R-DefineDBL**
- **R-InvokeDBL**
- **R-DropDBL**

f) **operationRequest:** the parameters of the indication primitive for this RDA operation.

g) **operationResult:** the parameters of the result response primitive for this RDA operation, which are present if the RDA operation has been successfully executed.

h) **operationError:** the parameters of the error response primitive for this RDA operation, which are present if the RDA operation has been unsuccessfully executed.

4.1.1.2 RDA dialogue entity

An RDA dialogue entity is created by a successful R-Initialize service, requesting initialization of an RDA dialogue, and exists until the RDA dialogue is terminated.

The RDA dialogue entity has the following attributes:

a) **dialogueID:** An identifier, which is unique within the OSI environment, used to identify the RDA dialogue.

b) **identityOfUser:** A name or other identifier known to the RDA client and the RDA server. It is used to identify the user, and may be used by the RDA server to determine privileges for the RDA dialogue. It consists of a character string.

NOTE – This attribute is used only by RDA Specializations.

c) **controlServicesAllowed:** A boolean attribute which indicates whether the RDA server has agreed to allow RDA Control services for this RDA dialogue to be issued from another RDA dialogue:

- **"false":** R-Cancel and R-Status requests for this RDA dialogue are not allowed from another RDA dialogue.
- **"true":** R-Cancel and R-Status requests for this RDA dialogue are allowed from another RDA dialogue.

d) **controlAuthenticationData:** Authentication data which is used to authenticate an R-Cancel or R-Status service request for this RDA dialogue when it is issued from another RDA dialogue to refer to this RDA dialogue.

e) **functionalUnitsAllowed:** The functional units successfully negotiated for this RDA dialogue:

- **"termination"**
- **"transaction"**
- **"cancel"**
- **"status"**
- **"resource"**
- **"immediate-dbl"**
- **"stored-dbl"**

f) **RDATransactionStatus:** The current state of the RDA dialogue with respect to RDA transactions:

- **RDATransactionNotOpen:** There is no RDA transaction currently in progress for the RDA dialogue.
- **RDATransactionOpen:** There is an RDA transaction currently in progress for the RDA dialogue, but it is not in the process of terminating.
- **RDATransactionTerminating:** There is an RDA transaction in progress for the RDA dialogue, but it is in the process of being terminated in order to commit or roll back the state of data resources.

g) **transactionRolledBack:** A boolean attribute which indicates whether or not the database server has rolled back the current RDA transaction:

- **"false":** The current transaction has not been rolled back.
- **"true":** The current transaction has been rolled back.

4.1.1.3 Opened data resource entity

An opened data resource entity is created by a successful R-Open service. An opened data resource entity is deleted by R-Close.

A data resource is always opened within the context of the RDA dialogue. It may also be opened within the context of another opened data resource, termed its parent. A data resource may have only one parent, and the data resource is automatically closed (and the opened data resource entity is deleted) if its parent is closed.

An opened data resource is also closed if the RDA dialogue to which it belongs is terminated.

The opened data resource entity has the following attributes:

a) **dialogueID:** The identifier for the RDA dialogue to which this opened data resource entity belongs.

b) **dataResourceHandle:** An integer that uniquely identifies this data resource within the set of data resources known to the RDA client within this RDA dialogue.

c) **parentDataResourceHandle:** The handle of the data resource referenced as parent when this data resource was opened, or "null".

d) **dataResourceName:** The name used to open the data resource. Data resources in the underlying database have names by which they are identified for opening purposes. It is possible for a RDA client to open a data resource without supplying a name, provided this is not ambiguous in that user's context.

4.1.1.4 Defined DBL entity

Each defined DBL entity represents one DBL statement that the RDA client is permitted to invoke within an RDA dialogue.

A defined DBL entity is created by a successful R-DefineDBL service request. A defined DBL entity is deleted independently of opened data resource entities by the R-DropDBL service.

Every defined DBL entity is related to an opened data resource entity, and is deleted from the RDA dialogue state if that opened data resource entity is deleted. A defined DBL entity is also deleted if the RDA dialogue to which it belongs is terminated.

The defined DBL entity has the following attributes:

a) **dialogueID:** The RDA dialogue identifier for the RDA dialogue to which this defined DBL entity belongs.

b) **commandHandle:** A unique identifier for defined DBL entities within this RDA dialogue.

c) **dataResourceHandle:** Identifies the opened data resource entity for the data resource accessed by the DBL statement of the R-DefineDBL service request.

4.1.2 General server execution rules

This subclause describes the general server execution rules which apply to all RDA service requests received by an RDA server. The general rules provide for specific types of augmentation based upon the particular RDA service request type. Those augmentations are defined in subsequent subclauses in this clause. Additional augmentations may be specified by individual application-contexts (in this part of ISO/IEC 9579) or by RDA Specializations (in other parts of ISO/IEC 9579).

4.1.2.1 Generation of the RDA operation entity

When an RDA service indication primitive is received by an RDA server it shall create an RDA operation entity with the following constraints on the initial attribute values:

Attribute	Initial value
dialogueID	If the type of the RDA service indication primitive identifies an R-Initialize service indication primitive, the dialogueID from that indication primitive; else the dialogueID from the RDA dialogue entity for the current RDA dialogue, if that entity exists; else "null".
operationID	The operationID parameter on the RDA service indication primitive.
status	awaitingExecution
cancelRequestReceived	"false"
operationType	The type of RDA service indication primitive.
operationRequest	The parameters on the RDA service indication primitive.
operationResult	"null"
operationError	"null"

NOTE – An RDA operation entity is not created if the RDA service-provider responds to or discards the RDA client request without issuing an indication primitive, as described in clause 4.2, RDA protocol machine.

After creation of the RDA operation entity, the following rules shall be applied to the newly created RDA operation entity:

If the operationType of the RDA operation entity identifies an RDA operation in a functional unit that is not listed in the functionalUnitsAllowed attribute of the RDA dialogue entity for this RDA dialogue, then the RDA operation entity shall be modified as follows:

Attribute	New value
status	aborted
operationError	serviceNotNegotiated

If there already exists another RDA operation entity with the same value for the dialogueID attribute and the same value for the operationID attribute, then the newly created RDA operation entity shall be modified as follows provided the value of its status attribute is awaitingExecution or executing:

Attribute	New value
status	aborted
operationError	duplicateOperationID

4.1.2.2 Implementor defined errors

When any implementor defined error occurs for an RDA operation entity which has a status attribute value of awaitingExecution or executing and whose operationType permits an error response of operationAborted, its RDA operation entity shall be modified as follows:

Attribute	New value
status	aborted
operationError	operationAborted together with supplementary information.

4.1.2.3 Beginning of an RDA operation

When an RDA operation which has a status attribute of awaitingExecution begins executing, its RDA operation entity shall be modified as follows:

Attribute	New value
status	executing

4.1.2.4 Cancellation of an RDA operation

If the RDA server cancels an RDA operation that has a status attribute value of executing, the following rule shall be applied:

Any RDA operation entity which has a status attribute value of executing and a cancelRequestReceived attribute of "true" shall be modified as follows:

Attribute	New value
status	cancelled
operationError	operationCancelled

If the RDA server sets the status attribute of an RDA operation entity to cancelled, then it shall ensure that the effect on any data resources referenced by that RDA operation is as if that RDA operation had never been executed.

4.1.2.5 Execution of an RDA operation

Any RDA operation entity which has a status attribute value of executing shall be modified according to the server execution rules for the particular operationType. The server execution rules for a particular RDA service include three general types of rules. These are

Entity manipulation rules: Entity manipulation rules specify constraints on the modification of the entities and attributes which comprise the RDA dialogue state. No modifications are allowed except when explicitly permitted.

Result rules: Result rules specify constraints on the result responses for particular RDA service request types.

Error rules: Error rules can define conditions which must exist in order for specific errors to be returned. Such rules are known as prerequisite predicates for the errors. If no prerequisite predicates are specified for a particular error in the context of a particular RDA operation, then issuance of the error is never prevented for that RDA operation by this part of ISO/IEC 9579.

Error rules can also define situations in which particular errors must occur. Such rules are known as mandatory predicates for the errors. If no mandatory predicates are specified for a particular error in the context of a particular RDA operation, then issuance of the error is never required for that RDA operation by this part of ISO/IEC 9579.

The following table summarizes the types of predicates which can be specified as error rules for specific errors in the context of particular RDA service request types:

Predicate type	Abbreviation for predicate type	Description
Mandatory	M	The specified error or some other error shall be returned if the predicate is "true".
Prerequisite	P	The specified error shall not be returned if the predicate is "false".

4.1.2.6 End of an RDA operation

When an RDA operation ends abnormally, an RDA operation entity whose status attribute is executing shall be modified as follows:

Attribute	New value
status	finished
operationError	Any error response value which is allowed for the request type indicated by the operationType attribute and which is not prohibited by any prerequisite error predicate for the request type.

When an RDA operation ends normally, an RDA operation entity whose status attribute is executing shall be modified as follows:

Attribute	New value
status	finished
operationResult	Any result response value which satisfies the constraints for the request type indicated by the operationType attribute provided no mandatory error predicate for the request type is satisfied.

4.1.2.7 Response to an RDA operation

An error response or result response primitive shall be issued only for an RDA operation entity which has a status attribute value of finished, cancelled, or aborted. The RDA operation entity shall be deleted when the response primitive is issued.

The order in which response primitives shall be issued is constrained by the following rules:

a) Following either an error response for an R-BeginTransaction operation or a transactionRolledBack error response for an R-ExecuteDBL or R-InvokeDBL operation, the RDA server shall discard any previously received indications. In addition, the RDA server shall delete any RDA operation entities created for those indications, and not issue response primitives for them.

b) Other than as stated above, the RDA server shall issue response primitives for R-Cancel and R-Status operations earlier than response primitives for other RDA operations.

c) The RDA server shall issue response primitives for all RDA operations other than R-Cancel and R-Status in the order in which the RDA operation entities were created.

The RDA server may execute more than one RDA operation at the same time. The RDA server shall ensure that the results of such execution are the same as they would have been had the RDA operations except R-Cancel and R-Status been executed serially in the order in which the RDA server received the RDA indication primitives.

4.1.2.8 Failure of the RDA dialogue

Following failure of an RDA dialogue:

- local recovery actions as appropriate for the application-context are performed (see 5.1.2.2, RDA dialogue failure, for the RDA Basic application-context; see ISO/IEC 10026-3 for the RDA TP application-context); and
- all entities (RDA operation, RDA dialogue, opened data resource, and defined DBL) whose dialogueID attribute equals the dialogueID for the current RDA dialogue shall be deleted.

4.1.3 RDA Dialogue Management services

4.1.3.1 RDA Dialogue Initialization functional unit

4.1.3.1.1 R-Initialize service

Entity manipulation rules

If an error is not returned then an RDA dialogue entity shall be created with the following constraints on the initial attribute values:

Attribute	Initial value
dialogueID	dialogueID attribute value of the RDA operation entity generated for this RDA operation.
identityOfUser	identityOfUser parameter value on the R-Initialize indication primitive.
controlServicesAllowed	"True" if the controlServiceDataRequested parameter value on the R-Initialize indication primitive is "true" and the RDA server permits other RDA dialogues to use RDA Control services on RDA operations of this RDA dialogue; "false" otherwise.
controlAuthenticationData	"null" if the controlServicesAllowed attribute is "false"; a value meaningful to the RDA server, if the controlServicesAllowed attribute is "true".
functionalUnitsAllowed	Functional units selected by the RDA server, which shall be a subset of or equal to those requested in the functionalUnitsRequested parameter on the R-Initialize indication primitive.
RDATransactionStatus	RDATransactionNotOpen
transactionRolledBack	"false"

Result rules

If a result is returned then the result parameters shall satisfy the following constraints:

Result parameter	Constraints
controlServicesAllowed	The value of the controlServicesAllowed attribute of the RDA dialogue entity for the current RDA dialogue if the value of the controlServiceDataRequested parameter of the R-Initialize indication primitive is "true". Otherwise this parameter shall not be present.
controlAuthenticationData	The value of the controlAuthenticationData attribute of the RDA dialogue entity for the current RDA dialogue if the value of the controlServicesAllowed attribute of the RDA dialogue entity for the current RDA dialogue is "true". Otherwise this parameter shall not be present.
functionalUnitsAllowed	The value of the functionalUnitsAllowed attribute of the RDA dialogue entity for the current RDA dialogue.

Error rules

An error shall be returned under the following conditions:

Error parameter	Type	Predicate
accessControlViolation	M+P	The RDA client does not have the required authority to execute an R-Initialize service request at the RDA server.
duplicateDialogueID	M+P	There exists an RDA dialogue entity whose dialogueID attribute equals the dialogueID parameter of the R-Initialize indication primitive.
userAuthenticationFailure	M+P	The identityOfUser parameter of the R-Initialize indication primitive is present and its value is not recognized by the RDA server, or the userAuthenticationData parameter of the R-Initialize indication primitive is present and its value failed to authenticate the user at the RDA server.

4.1.3.2 RDA Dialogue Termination functional unit

4.1.3.2.1 R-Terminate service

Entity manipulation rules

If an error is not returned then the following entities shall be deleted:

- all defined DBL entities which have a dialogueID attribute equal to the dialogueID for the current RDA dialogue;
- all opened data resource entities which have a dialogueID attribute equal to the dialogueID for the current RDA dialogue; and
- the RDA dialogue entity which has a dialogueID attribute equal to the dialogue ID for the current RDA dialogue.

Result rules

This part of ISO/IEC 9579 does not specify any result rules for this service.

Error rules

This part of ISO/IEC 9579 does not specify any additional error rules for this service.

4.1.4 RDA Transaction Management services

4.1.4.1 RDA Transaction Management functional unit

4.1.4.1.1 R-BeginTransaction service

Entity manipulation rules

The RDA dialogue entity for the current RDA dialogue shall be modified according to the following table:

Attribute	New value
RDATransactionStatus	RDATransactionOpen
transactionRolledBack	"false"

Result rules

There are no result responses for this service.

Error rules

This part of ISO/IEC 9579 does not specify any additional error rules for this service.

4.1.4.1.2 R-Commit service

Entity manipulation rules

The RDA dialogue entity for the current RDA dialogue shall be modified according to the following table:

Attribute	New value
RDATransactionStatus	RDATransactionNotOpen
transactionRolledBack	If the RDA transaction has been aborted and all changes made to data resources have been rolled back, then this attribute is set to "true".

Result rules

If a result is returned then the result parameters shall satisfy the following constraint:

Result parameter	Constraints
transactionResult	If the transactionRolledBack attribute of the RDA dialogue entity for the current RDA dialogue has the value "true", then this parameter shall have the value "rolledback". Otherwise, this parameter shall have the value "committed".

Error rules

This part of ISO/IEC 9579 does not specify any additional error rules for this service.

4.1.4.1.3 R-Rollback service

Entity manipulation rules

The RDA dialogue entity for the current RDA dialogue shall be modified according to the following table:

Attribute	New value
RDATransactionStatus	RDATransactionNotOpen
transactionRolledBack	"true"

Result rules

There are no result rules for this service.

Error rules

This part of ISO/IEC 9579 does not specify any additional error rules for this service.

4.1.5 RDA Control services

4.1.5.1 Cancel functional unit

4.1.5.1.1 R-Cancel service

Entity manipulation rules

Target RDA operation entities for an R-Cancel indication primitive are all those RDA operation entities for which

a) the dialogueID attribute is equal to the dialogueID parameter specified on the R-Cancel indication primitive, if such a parameter exists on the R-Cancel indication primitive; or

   the dialogueID attribute is equal to the dialogueID of the current RDA dialogue, if a dialogueID parameter does not exist on the R-Cancel indication primitive; and

b) the operationID attribute is equal to one of the values specified in the listOfOperationID parameter of the R-Cancel indication primitive, if such a parameter exists on the R-Cancel indication primitive; and

c) the RDA operation entity was created as the result of an indication primitive which was received before receipt of the R-Cancel indication primitive; and

d) the value of the operationType attribute is one of

   - R-Open,
   - R-Close,
   - R-ExecuteDBL,
   - R-DefineDBL,
   - R-InvokeDBL, or
   - R-DropDBL.

All target RDA operation entities whose status attribute is equal to awaitingExecution shall be modified according to the following table:

Attribute	New value
status	cancelled
cancelRequestReceived	"true"
operationError	operationCancelled

All target RDA operation entities whose status attribute is equal to executing shall be modified according to the following table:

Attribute	New value
cancelRequestReceived	"true"

NOTE – This part of ISO/IEC 9579 does not specify whether it is possible to cancel an RDA operation whose status is executing using the R-Cancel service.

Result rules

Any R-Cancel result shall be returned before any response is issued for an RDA operation entity whose cancelRequestReceived attribute was set to "true" as the result of the R-Cancel service.

Error rules

An error shall be returned under the following conditions:

Error parameter	Type	Predicate
controlAuthenticationFailure	M+P	The controlledDialogue parameter was specified on the R-Cancel indication primitive and there does not exist an RDA dialogue entity whose dialogueID attribute equals the dialogueID parameter of the R-Cancel indication primitive and whose controlAuthenticationData attribute equals the controlAuthenticationData parameter of the R-Cancel indication primitive.
controlServicesNotAllowed	M+P	The controlledDialogue parameter was specified on the R-Cancel indication primitive and there does not exist an RDA dialogue entity whose dialogueID attribute equals the dialogueID parameter of the R-Cancel indication primitive and whose controlServicesAllowed attribute is "true".
dialogueIDUnknown	M+P	The controlledDialogue parameter was specified on the R-Cancel indication primitive and there does not exist an RDA dialogue entity whose dialogueID attribute equals the dialogueID parameter of the R-Cancel indication primitive.

4.1.5.2 Status functional unit

4.1.5.2.1 R-Status service

Entity manipulation rules

There are no entity manipulation rules for this service.

Result rules

Target RDA operation entities for an R-Status indication primitive are all those RDA operation entities for which

a) the dialogueID attribute is equal to the dialogueID parameter specified on the R-Status indication primitive, if such a parameter exists on the R-Status indication primitive; or

the dialogueID attribute is equal to the dialogueID of the current RDA dialogue, if a dialogueID parameter does not exist on the R-Status indication primitive; and

b) the operationID attribute is equal to one of the values specified in the listOfOperationID parameter of the R-Status indication primitive, if such a parameter exists on the R-Status indication primitive; and

c) the RDA operation entity was created as the result of an indication primitive which was received before receipt of the R-Status indication primitive.

Any R-Status result shall be returned before any response is issued for a target RDA operation entity.

If a result is returned then the result parameters shall satisfy the following constraints:

Result parameter	Constraints
listOfStatusInformation	If the listOfOperationID parameter was specified in the R-Status indication primitive, then this parameter shall contain one item for each item in the listOfOperationID parameter. If the listOfOperationID parameter was not specified in the R-Status indication primitive, then this parameter shall contain one item for each target RDA operation entity.
operationID	If the listOfOperationID parameter was specified in the R-Status indication primitive, then the value of this parameter in the **n**th item of listOfStatusInformation shall equal the value of the **n**th item in the listOfOperationID parameter. If the listOfOperationID parameter was not specified in the R-Status indication primitive, then the value of this parameter in the **n**th item of listOfStatusInformation shall equal the value of the operationID attribute of the **n**th target RDA operation entity when the entities are ordered chronologically with the oldest being first.
operationStatus	The value of this parameter is constrained as specified for the different possible values below.
operationIDUnknown	There is no target RDA operation entity whose operationID attribute equals the operationID parameter for this listOfStatusInformation item.
awaitingExecution	There is a target RDA operation entity whose operationID attribute equals the operationID parameter for this listOfStatusInformation item and whose status attribute is equal to awaitingExecution.
executing	There is a target RDA operation entity whose operationID attribute equals the operationID parameter for this listOfStatusInformation item and whose status attribute is equal to executing.
finished	There is a target RDA operation entity whose operationID attribute equals the operationID parameter for this listOfStatusInformation item and whose status attribute is equal to finished.
cancelled	There is a target RDA operation entity whose operationID attribute equals the operationID parameter for this listOfStatusInformation item and whose status attribute is equal to cancelled.
aborted	There is a target RDA operation entity whose operationID attribute equals the operationID parameter for this listOfStatusInformation item and whose status attribute is equal to aborted.

Error rules

An error shall be returned under the following conditions:

Error parameter	Type	Predicate
controlAuthenticationFailure	M+P	The controlledDialogue parameter was specified on the R-Status indication primitive and there does not exist an RDA dialogue entity whose dialogueID attribute equals the dialogueID parameter of the R-Status indication primitive and whose controlAuthenticationData attribute equals the controlAuthenticationData parameter of the R-Status indication primitive.
controlServicesNotAllowed	M+P	The controlledDialogue parameter was specified on the R-Status indication primitive and there does not exist an RDA dialogue entity whose dialogueID attribute equals the dialogueID parameter of the R-Status indication primitive and whose controlServicesAllowed attribute is "true".
dialogueIDUnknown	M+P	The controlledDialogue parameter was specified on the R-Status indication primitive and there does not exist an RDA dialogue entity whose dialogueID attribute equals the dialogueID parameter of the R-Status indication primitive.

4.1.6 Resource Handling services

4.1.6.1 Resource Handling functional unit

4.1.6.1.1 R-Open service

Entity manipulation rules

If an error is not returned then an opened data resource entity shall be created with the following constraints on the initial attribute values:

Attribute	Initial value
dialogueID	The dialogueID for the current RDA dialogue.
dataResourceHandle	dataResourceHandle parameter value on the R-Open indication primitive.
parentDataResourceHandle	parentDataResourceHandle parameter value on the R-Open indication primitive.
dataResourceName	dataResourceName parameter value on the R-Open indication primitive.

Result rules

This part of ISO/IEC 9579 does not specify any result rules for this service.

Error rules

An error shall be returned under the following conditions:

Error parameter	Type	Predicate
dataResourceAlreadyOpen	P	There exists an opened data resource entity whose dialogueID attribute identifies the current RDA dialogue and whose dataResourceName equals the dataResourceName parameter of the R-Open indication primitive and whose parentDataResourceHandle equals the parentDataResourceHandle parameter of the R-Open indication primitive.
dataResourceNameNotSpecified	M+P	The dataResourceName parameter of the R-Open indication primitive is not present and more than one data resource is available in the current RDA dialogue at the RDA server.
dataResourceNotAvailable	M+P	The dataResourceName parameter of the R-Open indication primitive is present but no data resource with the specified name is available in the current RDA dialogue at the RDA server.
dataResourceUnknown	M+P	The dataResourceName parameter of the R-Open indication primitive is present but no data resource with the specified name is known at the RDA server.
duplicateDataResourceHandle	M+P	There exists an opened data resource entity whose dialogueID attribute identifies the current RDA dialogue and whose dataResourceHandle equals the dataResourceHandle parameter of the R-Open indication primitive.
parentDataResourceHandleUnknown	M+P	The parentDataResourceHandle parameter was specified on the R-Open indication primitive and there does not exist an opened data resource entity whose dialogueID attribute identifies the current RDA dialogue and whose dataResourceHandle equals the parentDataResourceHandle parameter of the R-Open indication primitive.

4.1.6.1.2 R-Close service

Entity manipulation rules

If an error is not returned then the following entities shall be deleted:

- all opened data resource entities which have a dialogueID attribute equal to the current RDA dialogue if the listOfDataResourceHandle parameter is not specified on the R-Close indication primitive;
- all opened data resource entities which have a dialogueID attribute equal to the current RDA dialogue and whose dataResourceHandle attribute is equal to an item in the listOfDataResourceHandle parameter in the R-Close indication primitive;
- all opened data resource entities which have a dialogueID attribute equal to the current dialogueID and whose parentDataResourceHandle attribute is equal to the dataResourceHandle attribute of an opened data resource entity which has been deleted in accordance with one of the rules in this subclause (including this rule); and
- all defined DBL entities which have a dialogueID attribute equal to the current dialogueID and whose dataResourceHandle attribute is equal to the dataResourceHandle attribute of an opened data resource entity which has been deleted in accordance with one of the rules in this subclause.

Result rules

If a result is returned then the result parameters shall satisfy the following constraints:

Result parameter	Constraints
dataResourceHandle	The value of this parameter shall be different in each item of listOfCloseExceptions. There shall exist an instance of this parameter whose value equals any value in the listOfDataResourceHandle on the R-Close indication primitive which does not match the dataResourceHandle attribute of an opened data resource entity deleted by the R-Close service. There shall not exist any instance of this parameter whose value is not equal to the dataResourceHandle attribute of an opened data resource entity deleted by this R-Close service. If the listOfDataResourceHandle was specified on the R-Close indication primitive, there shall not exist any instance of this parameter whose value is not a value in that listOfDataResourceHandle on the R-Close indication primitive.
dataResourceHandleUnknown	This parameter shall be specified only if the dataResourceHandle parameter in the same listOfCloseExceptions item does not specify the dataResourceHandle attribute of an opened data resource entity deleted by this R-Close service.
specificCloseException	This parameter shall be specified only if the dataResourceHandle parameter in the same listOfCloseExceptions item specifies the dataResourceHandle attribute of an opened data resource entity deleted by this R-Close service.

Error rules

This part of ISO/IEC 9579 does not specify any additional error rules for this service.

4.1.7 Database Language services

4.1.7.1 Immediate Execution DBL functional unit

4.1.7.1.1 R-ExecuteDBL service

Entity manipulation rules

If an error occurs that causes the RDA transaction to be rolled back, then the RDA dialogue entity for the current RDA dialogue shall be modified according to the following table:

Attribute	New value
transactionRolledBack	"true"

Result rules

If a result is returned then the result parameters shall satisfy the following constraints:

Result parameter	Constraints
listOfResultValues	The number of items in this parameter shall equal the repetitionCount parameter on the R-ExecuteDBL indication primitive if singleArgument was specified. The number of items in this parameter shall equal the number of items in the listOfSpecificDBLArgumentValues on the R-ExecuteDBL indication primitive if multipleArgument was specified. The number of items in this parameter shall be one if neither singleArgument nor multipleArgument was specified.

Error rules

An error shall be returned under the following conditions:

Error parameter	Type	Predicate
badRepetitionCount	M	The repetitionCount parameter of the R-ExecuteDBL indication primitive is not positive.
	P	The repetitionCount parameter of the R-ExecuteDBL indication primitive is not one.
dataResourceHandleNotSpecified	M+P	The dataResourceHandle parameter of the R-ExecuteDBL indication primitive is not present and more than one opened data resource entity exists whose dialogueID attribute identifies the current RDA dialogue.
dataResourceHandleUnknown	M+P	The dataResourceHandle parameter of the R-ExecuteDBL indication primitive is present and there does not exist an opened data resource entity whose dialogueID attribute identifies the current RDA dialogue and whose dataResourceHandle attribute equals the dataResourceHandle parameter of the R-ExecuteDBL indication primitive.
noDataResourceAvailable	M+P	The dataResourceHandle parameter of the R-ExecuteDBL indication primitive is not present and no opened data resource entity exists whose dialogueID attribute identifies the current RDA dialogue.
transactionRolledBack	M+P	The transactionRolledBack attribute of the RDA dialogue entity for the current RDA dialogue is "true".

4.1.7.2 Stored Execution DBL functional unit

4.1.7.2.1 R-DefineDBL service

Entity manipulation rules

If an error is not returned then a defined DBL entity shall be created with the following constraints on the initial attribute values:

Attribute	Initial value
dialogueID	The dialogueID for the current RDA dialogue.
commandHandle	commandHandle parameter value on the R-DefineDBL indication primitive.
dataResourceHandle	dataResourceHandle parameter value on the R-DefineDBL indication primitive.

Result rules

This part of ISO/IEC 9579 does not specify any result rules for this service.

Error rules

An error shall be returned under the following conditions:

Error parameter	Type	Predicate
dataResourceHandleNotSpecified	M+P	The dataResourceHandle parameter of the R-DefineDBL indication primitive is not present and more than one opened data resource entity exists whose dialogueID attribute identifies the current RDA dialogue.
dataResourceHandleUnknown	M+P	The dataResourceHandle parameter of the R-DefineDBL indication primitive is present and there does not exist an opened data resource entity whose dialogueID attribute identifies the current RDA dialogue and whose dataResourceHandle attribute equals the dataResourceHandle parameter of the R-DefineDBL indication primitive.
duplicateCommandHandle	M+P	There exists a defined DBL entity whose dialogueID attribute identifies the current RDA dialogue and whose commandHandle attribute equals the commandHandle parameter of the R-DefineDBL indication primitive.
noDataResourceAvailable	M+P	The dataResourceHandle parameter of the R-DefineDBL indication primitive is not present and no opened data resource entity exists whose dialogueID attribute identifies the current RDA dialogue.

4.1.7.2.2 R-InvokeDBL service

Entity manipulation rules

If an error occurs that causes the RDA transaction to be rolled back, then the RDA dialogue entity for the current RDA dialogue shall be modified according to the following table:

Attribute	New value
transactionRolledBack	"true"

Result rules

If a result is returned then the result parameters shall satisfy the following constraints:

Result parameter	Constraints
listOfResultValues	The number of items in this parameter shall equal the repetitionCount parameter on the R-InvokeDBL indication primitive if singleArgument was specified. The number of items in this parameter shall equal the number of items in the listOfSpecificDBLArgumentValues on the R-InvokeDBL indication primitive if multipleArgument was specified. The number of items in this parameter shall be one if neither singleArgument nor multipleArgument was specified.

Error rules

An error shall be returned under the following conditions:

Error parameter	Type	Predicate
badRepetitionCount	M	The repetitionCount parameter of the R-InvokeDBL indication primitive is not positive.
	P	The repetitionCount parameter of the R-InvokeDBL indication primitive is not one.
commandHandleUnknown	M+P	The commandHandle parameter of the R-InvokeDBL indication primitive is present and there does not exist a defined DBL entity whose dialogueID attribute identifies the current RDA dialogue and whose commandHandle attribute equals the commandHandle parameter of the R-InvokeDBL indication primitive.
transactionRolledBack	M+P	The transactionRolledBack attribute of the RDA dialogue entity for the current RDA dialogue is "true".

4.1.7.2.3 R-DropDBL service

Entity manipulation rules

If an error is not returned then all defined DBL entities which have a dialogueID attribute equal to the current RDA dialogue and whose commandHandle attribute is equal to an item in the listOfCommandHandle parameter in the R-DropDBL indication primitive shall be deleted.

Result rules

If a result is returned then the result parameters shall satisfy the following constraints:

Result parameter	Constraints
commandHandle	The value of this parameter shall be different in each item in the listOfDropDBLExceptions. There shall exist an instance of this parameter whose value equals any value in the listOfCommandHandle on the R-DropDBL indication primitive which does not match the commandHandle attribute of a defined DBL entity deleted by the R-DropDBL service. There shall not exist any instance of this parameter whose value is not a value in the listOfCommandHandle on the R-DropDBL indication primitive.
commandHandleUnknown	This parameter shall be specified only if the commandHandle parameter in the same listOfDropDBLExceptions item does not specify the commandHandle attribute of a defined DBL entity deleted by this R-DropDBL service.
specificDropDBLException	This parameter shall be specified only if the commandHandle parameter in the same listOfDropDBLExceptions item specifies the commandHandle attribute of a defined DBL entity deleted by this R-DropDBL service.

Error rules

This part of ISO/IEC 9579 does not specify any additional error rules for this service.

4.2 RDA protocol machine

This clause defines:

- the relationship of the functional units of the RDA Service to the functional units and APDUs of the RDA Protocol;
- the relationship of the names of RDA service primitives and their parameters to the names of RDA APDUs and their fields;
- the rules for concatenation of RDA APDUs;
- the state tables of the RDA protocol machine; and
- the protocol procedures of the RDA Protocol.

4.2.1 Functional units

The RDA Protocol functional units are equivalent to the functional units of the RDA Service. The RDA APDUs contained within each functional unit are specified in Table 17.

Table 17 – RDA functional units and associated RDA APDUs

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

4.2.2 Correspondence between RDA service primitives and RDA APDUs

The names of RDA APDUs have a direct mapping from the names of the corresponding RDA service primitives. The mapping is as follows:

- The name of an RDA APDU sent as a consequence of an RDA request service primitive is the name of the corresponding RDA service with "-RI" appended to it.

  NOTE 1 – For example, an R-Initialize-RI APDU is sent as a consequence of an R-Initialize request service primitive.

  NOTE 2 – There is no RDA service primitive associated with the R-Synchronize-RI APDU.

- The name of an RDA indication service primitive is the name of the corresponding RDA APDU with "-RI" removed from it.

  NOTE 3 – For example, an R-Initialize indication is generated as a consequence of the receipt of an R-Initialize-RI APDU.

- The name of an RDA APDU sent as a consequence of an RDA response service primitive is the name of the corresponding RDA service with "-RC" appended to it.

  NOTE 4 – For example, an R-Initialize-RC APDU is sent as a consequence of an R-Initialize response service primitive.

- The name of an RDA confirm service primitive is the name of the corresponding RDA APDU with "-RC" removed from it.

  NOTE 5 – For example, an R-Initialize confirm is generated as a consequence of the receipt of an R-Initialize-RC APDU.

The ASN.1 reference names of types (that is, field names) within RDA APDUs have a direct mapping from and to the names of the corresponding RDA service primitive parameters.

NOTE 6 – For example, the type with reference name functionalUnitsRequested within an R-Initialize-RI APDU maps from the functionalUnitsRequested parameter of the R-Initialize request service primitive.

The only field names that have such a mapping from and to RDA service parameters are those specified within RDA APDUs. That is, if a field name is not present in an APDU type but the correspondingly-named parameter is present in a corresponding service primitive, it is not mapped from or to the parameter of that service primitive.

NOTE 7 – For example, dialogueIDClientInvocation is not present in the R-Initialize-RI APDU (or in the R-Initialize request), but is present in the R-Initialize indication. Thus it is not mapped from the R-Initialize-RI APDU to the R-Initialize indication: rather, it is generated by the RDAPM of the RDA server.

In addition, if an RDA APDU is generated by the RDAPM directly – that is, the RDA APDU does not result from the issuance of the corresponding service primitive – and an RDA service primitive is issued upon receipt of that RDA APDU, then the mapping between the field names of that RDA APDU and the parameters of that RDA service primitive is only from the RDA APDU to the service primitive.

NOTE 8 – For example, when the RDAPM of the RDA server directly generates an "-RC" APDU for the invalidSequence error, the mapping is only from the field names of the "-RC" APDU to the corresponding parameters of the RDA confirm primitive.

4.2.3 Concatenation

Concatenation between RDA APDUs is not permitted.

4.2.4 State tables

This subclause defines two RDA protocol machines (RDAPMs), one for the RDA client and one for the RDA server, in terms of a set of state tables, collectively termed the RDAPM state table. The RDAPM state table specifies the interrelationship between the current state of an RDAPM, the incoming events that occur, and the resulting outgoing actions and RDAPM state.

Figure 4 illustrates the components of the RDAPM and the relationship of the RDAPM to the RDA model.

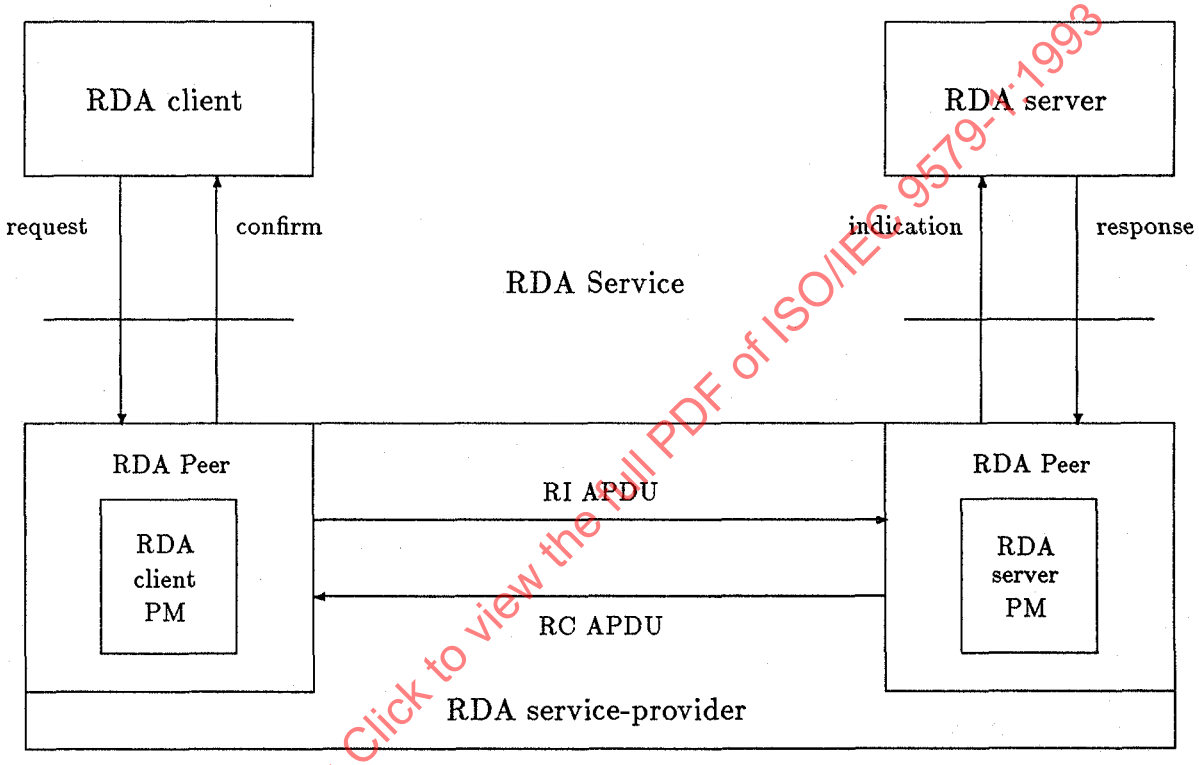

Figure 4 – Relationship of RDAPM to RDA model

Tables 18 to 32 specify the RDA protocol machine.

Tables 18 to 23 define the elements used in the RDAPM state table.

The overall RDAPM state table is divided into individual tables (Tables 24 to 31) for convenience and clarity. The individual state tables utilize the abbreviated names and identifiers of Tables 18 to 23.

4.2.4.1 Conventions

In the RDAPM state table, the intersection of an incoming event (row) and a state (column) forms a cell.

A non-blank cell represents a combination of an incoming event and a state that is defined for the RDAPM. A non-blank cell consists of one or more sub-cells. A sub-cell has an entry which consists of the following:

- zero, one, or more prerequisite predicate values;
- a resultant state;

- zero, one, or more outgoing actions; and
- zero, one, or more resultant predicate values.

At most one sub-cell of a non-blank cell has a set of prerequisite predicates that all evaluate to "true".

Each sub-cell is arranged as follows:

Prerequisite predicate value(s) Resultant state Outgoing action(s) Resultant predicate value(s)

An entry of logical not (¬) before a predicate indicates the logical negation of the predicate, as defined in Table 23.

An entry of asterisk (*) for the resultant state indicates that the state remains the same.

A blank cell represents a combination of an incoming event and state that is not defined for the RDAPM by this part of ISO/IEC 9579.

4.2.4.2 Actions to be taken by the RDAPM

The RDAPM state table defines the actions that shall be taken by the RDAPM upon receipt of an incoming event.

Prior to the successful initialization of an RDA dialogue on an association, the states of both the RDA client PM and the RDA server PM are set to inactive (I).

The state of the RDAPM is only changed as described in this clause or in particular application-contexts that define additional state changes.

A non-blank cell identifies a combination of an incoming event and a state for which RDA specifies an action. If such a combination occurs and either there are no prerequisite predicates or all of the prerequisite predicates of a sub-cell evaluate to "true", then the following actions are taken:

- the specified outgoing action or actions, if any, are performed;
- the state of the RDAPM is changed to the specified resultant state; and
- the resultant predicates are modified, if changes are required, so that they evaluate to "true".

A blank cell identifies a combination of an incoming event and a state for which RDA does not specify an action. If such a combination occurs, then

- If the incoming event is the receipt of an RDA primitive from the RDA service-user, then any resulting actions are a local matter and shall not result in an RDAPM state change or the flow of any RDA APDUs.
- If the incoming event is the receipt of an RDA APDU from an RDA peer, then any resulting state change or subsequent events are not defined by this part of ISO/IEC 9579.

When an association is released (normally or abnormally), both the RDA client PM and the RDA server PM cease to exist.

4.2.4.3 States

Table 18 specifies the identifier and description for each state.

Table 18 – States

State	Description
I	inactive
CB	RDA client PM pending R-Initialize-RC APDU
CA	RDA client dialogue active and no RDA transaction in progress
CT	RDA client dialogue active and RDA transaction in progress
CN	RDA client PM received transactionRolledBack error and R-Commit or R-Rollback request pending
CC	RDA client dialogue active and R-Commit-RC APDU pending
CR	RDA client dialogue active and R-Rollback-RC APDU pending
CE	RDA client PM pending R-Terminate-RC APDU
SB	RDA server PM pending R-Initialize response
SA	RDA server dialogue active and no RDA transaction in progress
ST	RDA server dialogue active and RDA transaction in progress
SN	RDA server PM sent transactionRolledBack error and R-Commit-RI or R-Rollback-RI APDU pending
SC	RDA server dialogue active and R-Commit response pending
SR	RDA server dialogue active and R-Rollback response pending
SS	RDA server PM sent R-BeginTransaction error and R-Synchronize-RI APDU pending
SE	RDA server PM pending R-Terminate response

4.2.4.4 Incoming events

Tables 19 and 20 specify the abbreviated name, source, and description for each incoming event.

The types of incoming events specified in Tables 19 and 20 are

- receipt of an RDA request primitive; or
- receipt of an RDA response primitive; or
- receipt of an RDA APDU as a presentation data value.

Table 19 – Incoming events: RDA Dialogue Management and RDA Transaction Management services

Event	Source	Description
R-Initialize req	RDA-user	R-Initialize request primitive issued by RDA client
R-Initialize-RI	RDA-peer	R-Initialize-RI APDU received by RDA server PM
R-Initialize rsp	RDA-user	R-Initialize response primitive issued by RDA server
R-Initialize-RC	RDA-peer	R-Initialize-RC APDU received by RDA client PM
R-Synchronize-RI	RDA-peer	R-Synchronize-RI APDU received by RDA server PM
R-Terminate req	RDA-user	R-Terminate request primitive issued by RDA client
R-Terminate-RI	RDA-peer	R-Terminate-RI APDU received by RDA server PM
R-Terminate rsp	RDA-user	R-Terminate response primitive issued by RDA server
R-Terminate-RC	RDA-peer	R-Terminate-RC APDU received by RDA client PM
R-BeginTransaction req	RDA-user	R-BeginTransaction request primitive issued by RDA client
R-BeginTransaction-RI	RDA-peer	R-BeginTransaction-RI APDU received by RDA server PM
R-BeginTransaction rsp	RDA-user	R-BeginTransaction response primitive issued by RDA server
R-BeginTransaction-RC	RDA-peer	R-BeginTransaction-RC APDU received by RDA client PM
R-Commit req	RDA-user	R-Commit request primitive issued by RDA client
R-Commit-RI	RDA-peer	R-Commit-RI APDU received by RDA server PM
R-Commit rsp	RDA-user	R-Commit response primitive issued by RDA server
R-Commit-RC	RDA-peer	R-Commit-RC APDU received by RDA client PM
R-Rollback req	RDA-user	R-Rollback request primitive issued by RDA client
R-Rollback-RI	RDA-peer	R-Rollback-RI APDU received by RDA server PM
R-Rollback rsp	RDA-user	R-Rollback response primitive issued by RDA server
R-Rollback-RC	RDA-peer	R-Rollback-RC APDU received by RDA client PM

Table 20 – Incoming events: RDA Control, Resource Handling, and Database Language services

Event	Source	Description
R-Cancel req	RDA-user	R-Cancel request primitive issued by RDA client
R-Cancel-RI	RDA-peer	R-Cancel-RI APDU received by RDA server PM
R-Cancel rsp	RDA-user	R-Cancel response primitive issued by RDA server
R-Cancel-RC	RDA-peer	R-Cancel-RC APDU received by RDA client PM
R-Status req	RDA-user	R-Status request primitive issued by RDA client
R-Status-RI	RDA-peer	R-Status-RI APDU received by RDA server PM
R-Status rsp	RDA-user	R-Status response primitive issued by RDA server
R-Status-RC	RDA-peer	R-Status-RC APDU received by RDA client PM
R-Open req	RDA-user	R-Open request primitive issued by RDA client
R-Open-RI	RDA-peer	R-Open-RI APDU received by RDA server PM
R-Open rsp	RDA-user	R-Open response primitive issued by RDA server
R-Open-RC	RDA-peer	R-Open-RC APDU received by RDA client PM
R-Close req	RDA-user	R-Close request primitive issued by RDA client
R-Close-RI	RDA-peer	R-Close-RI APDU received by RDA server PM
R-Close rsp	RDA-user	R-Close response primitive issued by RDA server
R-Close-RC	RDA-peer	R-Close-RC APDU received by RDA client PM
R-ExecuteDBL req	RDA-user	R-ExecuteDBL request primitive issued by RDA client
R-ExecuteDBL-RI	RDA-peer	R-ExecuteDBL-RI APDU received by RDA server PM
R-ExecuteDBL rsp	RDA-user	R-ExecuteDBL response primitive issued by RDA server
R-ExecuteDBL-RC	RDA-peer	R-ExecuteDBL-RC APDU received by RDA client PM
R-DefineDBL req	RDA-user	R-DefineDBL request primitive issued by RDA client
R-DefineDBL-RI	RDA-peer	R-DefineDBL-RI APDU received by RDA server PM
R-DefineDBL rsp	RDA-user	R-DefineDBL response primitive issued by RDA server
R-DefineDBL-RC	RDA-peer	R-DefineDBL-RC APDU received by RDA client PM
R-InvokeDBL req	RDA-user	R-InvokeDBL request primitive issued by RDA client
R-InvokeDBL-RI	RDA-peer	R-InvokeDBL-RI APDU received by RDA server PM
R-InvokeDBL rsp	RDA-user	R-InvokeDBL response primitive issued by RDA server
R-InvokeDBL-RC	RDA-peer	R-InvokeDBL-RC APDU received by RDA client PM
R-DropDBL req	RDA-user	R-DropDBL request primitive issued by RDA client
R-DropDBL-RI	RDA-peer	R-DropDBL-RI APDU received by RDA server PM
R-DropDBL rsp	RDA-user	R-DropDBL response primitive issued by RDA server
R-DropDBL-RC	RDA-peer	R-DropDBL-RC APDU received by RDA client PM

4.2.4.5 Outgoing actions

Tables 21 and 22 specify the identifier and description for each outgoing action.

The types of outgoing actions specified in Tables 21 and 22 are

- issuance of an RDA indication primitive; or
- issuance of an RDA confirm primitive; or
- sending of an RDA APDU as a presentation data value.

Table 21 – Outgoing actions: RDA Dialogue Management and RDA Transaction Management services

Action	Description
0	No action is performed on receipt of the incoming event in this state (that is, the incoming event is discarded)
1	Issue RC error response APDU with error "invalidSequence" and diagosticInformation as specified in Table 32
2	Send an R-Initialize-RI APDU
3	Issue an R-Initialize indication primitive
4	Send an R-Initialize-RC APDU
5	Issue an R-Initialize confirm primitive
6	Send an R-Synchronize-RI APDU
7	Send an R-Terminate-RI APDU
8	Issue an R-Terminate indication primitive
9	Send an R-Terminate-RC APDU
10	Issue an R-Terminate confirm primitive
11	Send an R-BeginTransaction-RI APDU
12	Issue an R-BeginTransaction indication primitive
13	Send an R-BeginTransaction-RC APDU
14	Issue an R-BeginTransaction confirm primitive
15	Send an R-Commit-RI APDU
16	Issue an R-Commit indication primitive
17	Send an R-Commit-RC APDU
18	Issue an R-Commit confirm primitive
19	Send an R-Rollback-RI APDU
20	Issue an R-Rollback indication primitive
21	Send an R-Rollback-RC APDU
22	Issue an R-Rollback confirm primitive

Table 22 – Outgoing actions: RDA Control, Resource Handling, and Database Language services

Action	Description
23	Send an R-Cancel-RI APDU
24	Issue an R-Cancel indication primitive
25	Send an R-Cancel-RC APDU
26	Issue an R-Cancel confirm primitive
27	Send an R-Status-RI APDU
28	Issue an R-Status indication primitive
29	Send an R-Status-RC APDU
30	Issue an R-Status confirm primitive
31	Send an R-Open-RI APDU
32	Issue an R-Open indication primitive
33	Send an R-Open-RC APDU
34	Issue an R-Open confirm primitive
35	Send an R-Close-RI APDU
36	Issue an R-Close indication primitive
37	Send an R-Close-RC APDU
38	Issue an R-Close confirm primitive
39	Send an R-ExecuteDBL-RI APDU
40	Issue an R-ExecuteDBL indication primitive
41	Send an R-ExecuteDBL-RC APDU
42	Issue an R-ExecuteDBL confirm primitive
43	Send an R-DefineDBL-RI APDU
44	Issue an R-DefineDBL indication primitive
45	Send an R-DefineDBL-RC APDU
46	Issue an R-DefineDBL confirm primitive
47	Send an R-InvokeDBL-RI APDU
48	Issue an R-InvokeDBL indication primitive
49	Send an R-InvokeDBL-RC APDU
50	Issue an R-InvokeDBL confirm primitive
51	Send an R-DropDBL-RI APDU
52	Issue an R-DropDBL indication primitive
53	Send an R-DropDBL-RC APDU
54	Issue an R-DropDBL confirm primitive

4.2.4.6 Predicates

Table 23 specifies the identifier for each predicate and the meaning of the predicate when the predicate evaluates to "true".

Table 23 – Predicates

Predicate	Meaning when true
TRB	The RDA server sent a transactionRolledBack error or the RDA client received a transaction-RolledBack error during the current RDA transaction.

4.2.4.7 RDAPM state tables

4.2.4.7.1 RDA client state tables

Table 24 specifies the state transitions and outgoing actions that occur in an RDA client PM for incoming events related to RDA Dialogue Management services.

Table 25 specifies the state transitions and outgoing actions that occur in an RDA client PM for incoming events related to RDA Transaction Management services.

Table 26 specifies the state transitions and outgoing actions that occur in an RDA client PM for incoming events related to RDA Control and Resource Handling services.

Table 27 specifies the state transitions and outgoing actions that occur in an RDA client PM for incoming events related to Database Language services.

Table 24 – RDA client state table: RDA Dialogue Management services

Incoming Event	state							
	I	CB	CA	CT	CN	CC	CR	CE
R-Initialize req	CB 2							
R-Initialize-RC (result response)		CA 5						
R-Initialize-RC (error response other than invalidSequence error)		I 5						
R-Initialize-RC (invalidSequence error)		* 5	* 5	* 5	* 5	* 5	* 5	* 5
R-Terminate req			CE 7					
R-Terminate-RC (result response)								I 10
R-Terminate-RC (error response other than invalidSequence error)								CA 10
R-Terminate-RC (invalidSequence error)		* 10		* 10	* 10	* 10	* 10	* 10

Table 25 – RDA client state table: RDA Transaction Management services

Incoming Event	State							
	I	CB	CA	CT	CN	CC	CR	CE
R-BeginTransaction req			CT 11 ¬ TRB					
R-BeginTransaction-RC (error response other than invalidSequence error)				CA 14 6		CA 14 6	CA 14 6	
R-BeginTransaction-RC (invalidSequence error)		* 14		* 14	* 14	* 14	* 14	* 14
R-Commit req				CC 15	CC 15			
R-Commit-RC (result response)						CA 18		
R-Commit-RC (error response other than invalidSequence error)						¬ TRB CT 18 TRB CN 18		
R-Commit-RC (invalidSequence error)		* 18	* 18			* 18	* 18	* 18
R-Rollback req				CR 19	CR 19			
R-Rollback-RC (result response)							CA 22	
R-Rollback-RC (error response other than invalidSequence error)							¬ TRB CT 22 TRB CN 22	
R-Rollback-RC (invalidSequence error)		* 22	* 22			* 22	* 22	* 22

Table 26 – RDA client state table: RDA Control and Resource Handling services

Incoming Event	State							
	I	CB	CA	CT	CN	CC	CR	CE
R-Cancel req			* 23	* 23				
R-Cancel-RC (result response, or error response other than invalidSequence error)			* 26	* 26		* 26	* 26	* 26
R-Cancel-RC (invalidSequence error)		* 26				* 26	* 26	* 26
R-Status req			* 27	* 27				
R-Status-RC (result response, or error response other than invalidSequence error)			* 30	* 30		* 30	* 30	* 30
R-Status-RC (invalidSequence error)		* 30				* 30	* 30	* 30
R-Open req			* 31	* 31				
R-Open-RC (result response, or error response other than invalidSequence error)			* 34	* 34		* 34	* 34	* 34
R-Open-RC (invalidSequence error)		* 34				* 34	* 34	* 34
R-Close req			* 35	* 35				
R-Close-RC (result response, or error response other than invalidSequence error)			* 38	* 38		* 38	* 38	* 38
R-Close-RC (invalidSequence error)		* 38				* 38	* 38	* 38

Table 27 – RDA client state table: Database Language services

Incoming Event	State							
	I	CB	CA	CT	CN	CC	CR	CE
R-ExecuteDBL req			* 39	* 39				
R-ExecuteDBL-RC (result response, or error response other than transactionRolledBack or invalidSequence error)			* 42	* 42		* 42	* 42	* 42
R-ExecuteDBL-RC (transactionRolledBack error)				CN 42 TRB		* 42 TRB	* 42 TRB	
R-ExecuteDBL-RC (invalidSequence error)		* 42				* 42	* 42	* 42
R-DefineDBL req			* 43	* 43				
R-DefineDBL-RC (result response, or error response other than invalidSequence error)			* 46	* 46		* 46	* 46	* 46
R-DefineDBL-RC (invalidSequence error)		* 46				* 46	* 46	* 46
R-InvokeDBL req			* 47	* 47				
R-InvokeDBL-RC (result response, or error response other than transactionRolledBack or invalidSequence error)			* 50	* 50		* 50	* 50	* 50
R-InvokeDBL-RC (transactionRolledBack error)				CN 50 TRB		* 50 TRB	* 50 TRB	
R-InvokeDBL-RC (invalidSequence error)		* 50				* 50	* 50	* 50
R-DropDBL req			* 51	* 51				
R-DropDBL-RC (result response, or error response other than invalidSequence error)			* 54	* 54		* 54	* 54	* 54
R-DropDBL-RC (invalidSequence error)		* 54				* 54	* 54	* 54

4.2.4.7.2 RDA server state tables

Table 28 specifies the state transitions and outgoing actions that occur in an RDA server PM for incoming events related to RDA Dialogue Management services.

Table 29 specifies the state transitions and outgoing actions that occur in an RDA server PM for incoming events related to RDA Transaction Management services.

Table 30 specifies the state transitions and outgoing actions that occur in an RDA server PM for incoming events related to RDA Control and Resource Handling services.

Table 31 specifies the state transitions and outgoing actions that occur in an RDA server PM for incoming events related to Database Language services.

Table 28 – RDA server state table: RDA Dialogue Management services

Incoming Event	State								
	I	SB	SA	ST	SN	SC	SR	SS	SE
R-Initialize-RI	SB 3	* 1	* 1	* 1	* 0	* 1	* 1	* 0	* 1
R-Initialize rsp (result response)		SA 4							
R-Initialize rsp (error response)		I 4							
R-Synchronize-RI								SA	
R-Terminate-RI	* 1	* 1	SE 8	* 1	* 0	* 1	* 1	* 0	* 1
R-Terminate rsp (result response)									I 9
R-Terminate rsp (error response)									SA 9

Table 29 – RDA server state table: RDA Transaction Management services

Incoming Event	State								
	I	SB	SA	ST	SN	SC	SR	SS	SE
R-BeginTransaction-RI	* 1	* 1	ST 12 ¬ TRB	* 1	* 0	* 1	* 1	* 0	* 1
R-BeginTransaction rsp (error response)				SS 13		SS 13	SS 13		
R-Commit-RI	* 1	* 1	* 1	SC 16	SC 16	* 1	* 1	* 0	* 1
R-Commit rsp (result response)						SA 17			
R-Commit rsp (error response)						¬ TRB ST 17 TRB SN 17			
R-Rollback-RI	* 1	* 1	* 1	SR 20	SR 20	* 1	* 1	* 0	* 1
R-Rollback rsp (result response)							SA 21		
R-Rollback-RC (error response)							¬ TRB ST 21 TRB SN 21		

Table 30 – RDA server state table: RDA Control and Resource Handling services

Incoming Event	State								
	I	SB	SA	ST	SN	SC	SR	SS	SE
R-Cancel-RI	* 1	* 1	* 24	* 24	* 0	* 1	* 1	* 0	* 1
R-Cancel rsp			* 25	* 25		* 25	* 25		* 25
R-Status-RI	* 1	* 1	* 28	* 28	* 0	* 1	* 1	* 0	* 1
R-Status rsp			* 29	* 29		* 29	* 29		* 29
R-Open-RI	* 1	* 1	* 32	* 32	* 0	* 1	* 1	* 0	* 1
R-Open rsp			* 33	* 33		* 33	* 33		* 33
R-Close-RI	* 1	* 1	* 36	* 36	* 0	* 1	* 1	* 0	* 1
R-Close rsp			* 37	* 37		* 37	* 37		* 37

Table 31 – RDA server state table: Database Language services

Incoming Event	State								
	I	SB	SA	ST	SN	SC	SR	SS	SE
R-ExecuteDBL-RI	* 1	* 1	* 40	* 40	* 0	* 1	* 1	* 0	* 1
R-ExecuteDBL rsp (response other than transactionRolledBack error)			* 41	* 41		* 41	* 41		* 41
R-ExecuteDBL rsp (transactionRolledBack error)				SN 41 TRB		* 41 TRB	* 41 TRB		
R-DefineDBL-RI	* 1	* 1	* 44	* 44	* 0	* 1	* 1	* 0	* 1
R-DefineDBL rsp			* 45	* 45		* 45	* 45		* 45
R-InvokeDBL-RI	* 1	* 1	* 48	* 48	* 0	* 1	* 1	* 0	* 1
R-InvokeDBL rsp (response other than transactionRolledBack error)			* 49	* 49		* 49	* 49		* 49
R-InvokeDBL rsp (transactionRolledBack error)				SN 49 TRB		* 49 TRB	* 49 TRB		
R-DropDBL-RI	* 1	* 1	* 52	* 52	* 0	* 1	* 1	* 0	* 1
R-DropDBL rsp			* 53	* 53		* 53	* 53		* 53

4.2.4.7.3 Values of diagnosticInformation for invalidSequence error

Table 32 specifies the value of the diagnosticInformation parameter for each situation in which an invalidSequence error can occur.

Table 32 – diagnosticInformation for invalidSequence error

Incoming Event	State								
	I	SB	SA	ST	SN	SC	SR	SS	SE
R-Initialize-RI		E2	E3	E3		E3	E3		E7
R-Synchronize-RI									
R-Terminate-RI	E1	E2		E5		E6	E6		E7
R-BeginTransaction-RI	E1	E2		E5		E6	E6		E7
R-Commit-RI	E1	E2	E4			E6	E6		E7
R-Rollback-RI	E1	E2	E4			E6	E6		E7
R-Cancel-RI	E1	E2				E6	E6		E7
R-Status-RI	E1	E2				E6	E6		E7
R-Open-RI	E1	E2				E6	E6		E7
R-Close-RI	E1	E2				E6	E6		E7
R-ExecuteDBL-RI	E1	E2				E6	E6		E7
R-DefineDBL-RI	E1	E2				E6	E6		E7
R-InvokeDBL-RI	E1	E2				E6	E6		E7
R-DropDBL-RI	E1	E2				E6	E6		E7

Definitions of diagnosticInformation errors:

E1 RDA dialogue not active
E2 RDA dialogue initializing
E3 RDA dialogue already active
E4 RDA transaction not open
E5 RDA transaction open
E6 RDA transaction terminating
E7 RDA dialogue terminating

4.2.5 Protocol procedures

4.2.5.1 Initialization of an RDA dialogue

An RDA dialogue is initialized by the RDAPM of the RDA server upon sending of an R-Initialize-RC result response APDU. Otherwise, the RDA dialogue is not initialized. When initialized, the RDA dialogue enters state SA.

An RDA dialogue is initialized by the RDAPM of the RDA client upon receipt of an R-Initialize-RC result response APDU. Otherwise, the RDA dialogue is not initialized. When initialized, the RDA dialogue enters state CA.

4.2.5.2 Termination of an RDA dialogue

An RDA dialogue is terminated by the RDAPM of the RDA server upon sending of an R-Terminate-RC result response APDU. Otherwise, the RDA dialogue is not terminated.

An RDA dialogue is terminated by the RDAPM of the RDA client upon receipt of an R-Terminate-RC result response APDU. Otherwise, the RDA dialogue is not terminated.

4.2.5.3 Initiation of an RDA transaction

An RDA transaction is initiated by the RDAPM of the RDA client upon sending of an R-BeginTransaction-RI APDU.

An RDA transaction is initiated by the RDAPM of the RDA server upon receipt of an R-BeginTransaction-RI APDU.

4.2.5.4 Termination of an RDA transaction

An RDA transaction is terminated by the RDAPM of the RDA server upon sending of either

- an R-BeginTransaction-RC APDU;
- an R-Commit-RC result response APDU; or
- an R-Rollback-RC result response APDU.

An RDA transaction is not terminated by the RDAPM of the RDA server upon sending of either an R-Commit-RC error response APDU or an R-Rollback-RC error response APDU.

An RDA transaction is terminated by the RDAPM of the RDA client upon receipt of either

- an R-BeginTransaction-RC APDU;
- an R-Commit-RC result response APDU; or
- an R-Rollback-RC result response APDU.

An RDA transaction is not terminated by the RDAPM of the RDA client upon receipt of either an R-Commit-RC error response APDU or an R-Rollback-RC error response APDU.

4.3 Application-protocol-data-units

This clause defines the structure of the RDA application-protocol-data-units in the form of a template ASN.1 module, which each RDA Specialization shall complete as specified later in this clause (supplying a type definition for each parameter whose type is left unspecified, possibly deleting OPTIONAL parameters or making them mandatory, and assigning application tags where necessary).

NOTE – The RDA definitions have names beginning with "R-". In accordance with ASN.1 conventions, local names and names of values start with a lower case letter and names of types start with an upper case letter.

```
ISO9579-RDAxxx { iso standard rda(9579) part-n(n) module(0) version-m(m) }

--The "xxx" in "RDAxxx" shall be chosen by the RDA Specialization.
--The "n" in "part-n(n)" shall be the same as that for the RDA Specialization.
--The "m" in "version-m(m)" shall be chosen by the RDA Specialization.

DEFINITIONS IMPLICIT TAGS ::= BEGIN

IMPORTS   AP-title,
          AE-qualifier,
          AP-invocation-identifier,
          AE-invocation-identifier
               FROM ACSE-1 { joint-iso-ccitt standard acse(8650) } ;

--top level RDA APDU CHOICE:

RDA-APDU                    ::= CHOICE
  { r-Initialize-RI                  [0] R-Initialize-RI,
    r-Initialize-RC                  [1] R-Initialize-RC,
    r-Synchronize-RI                 [2] R-Synchronize-RI,
    r-Terminate-RI                   [3] R-Terminate-RI,
    r-Terminate-RC                   [4] R-Terminate-RC,
    r-BeginTransaction-RI            [5] R-BeginTransaction-RI,
    r-BeginTransaction-RC            [6] R-BeginTransaction-RC,
    r-Commit-RI                      [7] R-Commit-RI,
    r-Commit-RC                      [8] R-Commit-RC,
    r-Rollback-RI                    [9] R-Rollback-RI
    r-Rollback-RC                   [10] R-Rollback-RC,
    r-Cancel-RI                     [11] R-Cancel-RI,
    r-Cancel-RC                     [12] R-Cancel-RC,
    r-Status-RI                     [13] R-Status-RI,
    r-Status-RC                     [14] R-Status-RC,
    r-Open-RI                       [15] R-Open-RI,
    r-Open-RC                       [16] R-Open-RC,
    r-Close-RI                      [17] R-Close-RI,
    r-Close-RC                      [18] R-Close-RC,
    r-ExecuteDBL-RI                 [19] R-ExecuteDBL-RI,
    r-ExecuteDBL-RC                 [20] R-ExecuteDBL-RC,
    r-DefineDBL-RI                  [21] R-DefineDBL-RI,
    r-DefineDBL-RC                  [22] R-DefineDBL-RC,
    r-InvokeDBL-RI                  [23] R-InvokeDBL-RI,
    r-InvokeDBL-RC                  [24] R-InvokeDBL-RC,
```

```
    r-DropDBL-RI                    [25] R-DropDBL-RI,
    r-DropDBL-RC                    [26] R-DropDBL-RC
  }

-- Individual RDA APDU definitions:

R-Initialize-RI             ::= SEQUENCE
  { operationID                         OperationID,
    r-Initialize-req                [0] R-Initialize-Request
  }

R-Initialize-RC             ::= SEQUENCE
  { operationID                         OperationID,
    res-or-err                      CHOICE
    { r-Initialize-res              [0] R-Initialize-Result,
      r-Initialize-err              [1] R-Initialize-Error
    }
  }

R-Synchronize-RI            ::= SEQUENCE
  {
  }

R-Terminate-RI              ::= SEQUENCE
  { operationID                         OperationID,
    r-Terminate-req                 [0] R-Terminate-Request
  }

R-Terminate-RC              ::= SEQUENCE
  { operationID                         OperationID,
    res-or-err                      CHOICE
    { r-Terminate-res               [0] R-Terminate-Result,
      r-Terminate-err               [1] R-Terminate-Error
    }
  }

R-BeginTransaction-RI       ::= SEQUENCE
  { operationID                         OperationID,
    r-BeginTransaction-req          [0] NULL
  }

R-BeginTransaction-RC       ::= SEQUENCE
  { operationID                         OperationID,
    r-BeginTransaction-err          [0] R-BeginTransaction-Error
  }

R-Commit-RI                 ::= SEQUENCE
  { operationID                         OperationID,
    r-Commit-req                    [0] NULL
  }
```

```
R-Commit-RC                ::= SEQUENCE
  { operationID                    OperationID,
    res-or-err               CHOICE
    { r-Commit-res            [0] R-Commit-Result,
      r-Commit-err            [1] R-Commit-Error
    }
  }

R-Rollback-RI              ::= SEQUENCE
  { operationID                    OperationID,
    r-Rollback-req           [0] NULL
  }

R-Rollback-RC              ::= SEQUENCE
  { operationID                    OperationID,
    res-or-err               CHOICE
    { r-Rollback-res          [0] NULL
      r-Rollback-err          [1] R-Rollback-Error
    }
  }

R-Cancel-RI                ::= SEQUENCE
  { operationID                    OperationID,
    r-Cancel-req             [0] R-Cancel-Request
  }

R-Cancel-RC                ::= SEQUENCE
  { operationID                    OperationID,
    res-or-err               CHOICE
    { r-Cancel-res            [0] R-Cancel-Result,
      r-Cancel-err            [1] R-Cancel-Error
    }
  }

R-Status-RI                ::= SEQUENCE
  { operationID                    OperationID,
    r-Status-req             [0] R-Status-Request
  }

R-Status-RC                ::= SEQUENCE
  { operationID                    OperationID,
    res-or-err               CHOICE
     { r-Status-res           [0] R-Status-Result,
       r-Status-err           [1] R-Status-Error
    }
  }

R-Open-RI                  ::= SEQUENCE
  { operationID                    OperationID,
    r-Open-req               [0] R-Open-Request
  }
```

```
R-Open-RC                    ::= SEQUENCE
  { operationID                        OperationID,
    res-or-err                       CHOICE
    { r-Open-res                      [0] R-Open-Result,
      r-Open-err                      [1] R-Open-Error
    }
  }

R-Close-RI                   ::= SEQUENCE
  { operationID                        OperationID,
    r-Close-req                     [0] R-Close-Request
  }

R-Close-RC                   ::= SEQUENCE
  { operationID                        OperationID,
    res-or-err                       CHOICE
    { r-Close-res                     [0] R-Close-Result,
      r-Close-err                     [1] R-Close-Error
    }
  }

R-ExecuteDBL-RI              ::= SEQUENCE
  { operationID                        OperationID,
    r-ExecuteDBL-req                [0] R-ExecuteDBL-Request
  }

R-ExecuteDBL-RC              ::= SEQUENCE
  { operationID                        OperationID,
    res-or-err                       CHOICE
    { r-ExecuteDBL-res                [0] R-ExecuteDBL-Result,
      r-ExecuteDBL-err                [1] R-ExecuteDBL-Error
    }
  }

R-DefineDBL-RI               ::= SEQUENCE
  { operationID                        OperationID,
    r-DefineDBL-req                 [0] R-DefineDBL-Request
  }

R-DefineDBL-RC               ::= SEQUENCE
  { operationID                        OperationID,
    res-or-err                       CHOICE
    { r-DefineDBL-res                 [0] R-DefineDBL-Result,
      r-DefineDBL-err                 [1] R-DefineDBL-Error
    }
  }

R-InvokeDBL-RI               ::= SEQUENCE
  { operationID                        OperationID,
    r-InvokeDBL-req                 [0] R-InvokeDBL-Request
  }
```

```
R-InvokeDBL-RC              ::= SEQUENCE
  { operationID                       OperationID,
    res-or-err                  CHOICE
    { r-InvokeDBL-res            [0] R-InvokeDBL-Result,
      r-InvokeDBL-err            [1] R-InvokeDBL-Error
    }
  }

R-DropDBL-RI                ::= SEQUENCE
  { operationID                       OperationID,
    r-DropDBL-req              [0] R-DropDBL-Request
  }

R-DropDBL-RC                ::= SEQUENCE
  { operationID                       OperationID,
    res-or-err                  CHOICE
    { r-DropDBL-res              [0] R-DropDBL-Result,
      r-DropDBL-err              [1] R-DropDBL-Error
    }
  }

R-Initialize-Request        ::= SEQUENCE
  { dialogueIDSuffix           [0] DialogueIDSuffix,
    identityOfUser             [1] VisibleString,
    userAuthenticationData     [2] AuthenticationData OPTIONAL,
    controlServiceDataRequested [3] BOOLEAN DEFAULT FALSE,
    functionalUnitsRequested   [4] FunctionalUnits,
    specificInitializeArgument [30] SpecificInitializeArgument OPTIONAL
  }

R-Initialize-Result         ::= SEQUENCE
  { controlServiceData         [0] SEQUENCE
    { controlServicesAllowed     [0] BOOLEAN DEFAULT TRUE,
      controlAuthenticationData  [1] AuthenticationData OPTIONAL
    } OPTIONAL,
    functionalUnitsAllowed     [1] FunctionalUnits,
    specificInitializeResult   [30] SpecificInitializeResult OPTIONAL
  }

R-Initialize-Error          ::= CHOICE
  { accessControlViolation          AccessControlViolation,
    duplicateDialogueID             DuplicateDialogueID,
    invalidSequence                 InvalidSequence,
    operationAborted                OperationAborted,
    userAuthenticationFailure       UserAuthenticationFailure,
    specificInitializeError         SpecificInitializeError
  }
```

```
R-Terminate-Request          ::= SEQUENCE
  { specificTerminateArgument        [30] SpecificTerminateArgument OPTIONAL
  }

R-Terminate-Result           ::= SEQUENCE
  { specificTerminateResult          [30] SpecificTerminateResult OPTIONAL
  }

R-Terminate-Error            ::= CHOICE
  { duplicateOperationID                  DuplicateOperationID,
    invalidSequence                       InvalidSequence,
    operationAborted                      OperationAborted,
    serviceNotNegotiated                  ServiceNotNegotiated,
    specificTerminateError                SpecificTerminateError
  }

R-BeginTransaction-Error     ::= CHOICE
  { duplicateOperationID                  DuplicateOperationID,
    invalidSequence                       InvalidSequence,
    operationAborted                      OperationAborted,
    serviceNotNegotiated                  ServiceNotNegotiated
  }

R-Commit-Result              ::= SEQUENCE
  { transactionResult                [0] ENUMERATED
    { committed                           (0),
      rolledBack                          (1)
    }
  }

R-Commit-Error               ::= CHOICE
  { duplicateOperationID                  DuplicateOperationID,
    invalidSequence                       InvalidSequence
  }

R-Rollback-Error             ::= CHOICE
  { duplicateOperationID                  DuplicateOperationID,
    invalidSequence                       InvalidSequence
  }

R-Cancel-Request             ::= SEQUENCE
  { controlledDialogue               [0] SEQUENCE
    { dialogueIDClientInvocation       [0] DialogueIDClientInvocation OPTIONAL,
      dialogueIDSuffix                 [1] DialogueIDSuffix,
      controlAuthenticationData        [2] AuthenticationData
    } OPTIONAL,
    listOfOperationID                [1] SEQUENCE OF OperationID OPTIONAL,
    specificCancelArgument           [30] SpecificCancelArgument OPTIONAL
  }
```

```
R-Cancel-Result             ::= SEQUENCE
  { specificCancelResult            [30] SpecificCancelResult OPTIONAL
  }

R-Cancel-Error              ::= CHOICE
  { controlAuthenticationFailure        ControlAuthenticationFailure,
    controlServicesNotAllowed           ControlServicesNotAllowed,
    dialogueIDUnknown                   DialogueIDUnknown,
    duplicateOperationID                DuplicateOperationID,
    invalidSequence                     InvalidSequence,
    operationAborted                    OperationAborted,
    serviceNotNegotiated                ServiceNotNegotiated,
    specificCancelError                 SpecificCancelError
  }

R-Status-Request            ::= SEQUENCE
  { controlledDialogue              [0] SEQUENCE
    { dialogueIDClientInvocation      [0] DialogueIDClientInvocation OPTIONAL,
      dialogueIDSuffix                [1] DialogueIDSuffix,
      controlAuthenticationData       [2] AuthenticationData
    } OPTIONAL,
    listOfOperationID               [1] SEQUENCE OF OperationID OPTIONAL,
    specificStatusArgument         [30] SpecificStatusArgument OPTIONAL
  }

R-Status-Result             ::= SEQUENCE
  { listOfStatusInformation         [0] SEQUENCE OF StatusInformation OPTIONAL,
    specificStatusResult           [30] SpecificStatusResult OPTIONAL
  }

R-Status-Error              ::= CHOICE
  { controlAuthenticationFailure        ControlAuthenticationFailure,
    controlServicesNotAllowed           ControlServicesNotAllowed,
    dialogueIDUnknown                   DialogueIDUnknown,
    duplicateOperationID                DuplicateOperationID,
    invalidSequence                     InvalidSequence,
    operationAborted                    OperationAborted,
    serviceNotNegotiated                ServiceNotNegotiated,
    specificStatusError                 SpecificStatusError
  }

R-Open-Request              ::= SEQUENCE
  { dataResourceHandle              [0] DataResourceHandle,
    parentDataResourceHandle        [1] DataResourceHandle OPTIONAL,
    dataResourceName                [2] VisibleString OPTIONAL,
    specificAccessControlData       [3] SpecificAccessControlData OPTIONAL,
    specificUsageMode               [4] SpecificUsageMode OPTIONAL,
    specificOpenArgument           [30] SpecificOpenArgument OPTIONAL
  }
```

```
R-Open-Result                ::= SEQUENCE
  { specificOpenResult               [30] SpecificOpenResult OPTIONAL
  }

R-Open-Error                 ::= CHOICE
  { dataResourceAlreadyOpen                DataResourceAlreadyOpen,
    dataResourceNameNotSpecified           DataResourceNameNotSpecified,
    dataResourceNotAvailable               DataResourceNotAvailable,
    dataResourceUnknown                    DataResourceUnknown,
    duplicateDataResourceHandle            DuplicateDataResourceHandle,
    duplicateOperationID                   DuplicateOperationID,
    invalidSequence                        InvalidSequence,
    operationAborted                       OperationAborted,
    operationCancelled                     OperationCancelled,
    parentDataResourceHandleUnknown        ParentDataResourceHandleUnknown,
    serviceNotNegotiated                   ServiceNotNegotiated,
    specificOpenError                      SpecificOpenError
  }

R-Close-Request              ::= SEQUENCE
  { listOfDataResourceHandle          [0] SEQUENCE OF DataResourceHandle OPTIONAL,
    specificCloseArgument            [30] SpecificCloseArgument OPTIONAL
  }

R-Close-Result               ::= SEQUENCE
  { listOfCloseExceptions             [0] SEQUENCE OF CloseException OPTIONAL,
    specificCloseResult              [30] SpecificCloseResult OPTIONAL
  }

R-Close-Error                ::= CHOICE
  { duplicateOperationID                   DuplicateOperationID,
    invalidSequence                        InvalidSequence,
    operationAborted                       OperationAborted,
    operationCancelled                     OperationCancelled,
    serviceNotNegotiated                   ServiceNotNegotiated,
    specificCloseError                     SpecificCloseError
  }

R-ExecuteDBL-Request         ::= SEQUENCE
  { dataResourceHandle                [0] DataResourceHandle OPTIONAL,
    specificDBLStatement              [1] SpecificDBLStatement,
    specificDBLArgumentSpecification  [2] SpecificDBLArgumentSpecification OPTIONAL,
    specificDBLResultSpecification    [3] SpecificDBLResultSpecification OPTIONAL,
    dblArguments                     CHOICE
    { singleArgument                  [4] SEQUENCE
      { repetitionCount                 [0] INTEGER DEFAULT 1,
        specificDBLArgumentValues       [1] SpecificDBLArgumentValues OPTIONAL
      },
```

```
      multipleArgument                [5]   SEQUENCE
      { listOfSpecificDBLArgumentValues [0]   SEQUENCE OF SpecificDBLArgumentValues
      }
    } OPTIONAL
  }

R-ExecuteDBL-Result          ::= SEQUENCE
  { specificTerminationCode           [0] SpecificTerminationCode OPTIONAL,
    specificDBLResultSpecification    [1] SpecificDBLResultSpecification OPTIONAL,
    listOfResultValues                [2] SEQUENCE OF ResultValues OPTIONAL
  }

R-ExecuteDBL-Error           ::= CHOICE
  { badRepetitionCount                    BadRepetitionCount,
    dataResourceHandleNotSpecified        DataResourceHandleNotSpecified,
    dataResourceHandleUnknown             DataResourceHandleUnknown,
    duplicateOperationID                  DuplicateOperationID,
    invalidSequence                       InvalidSequence,
    noDataResourceAvailable               NoDataResourceAvailable,
    operationAborted                      OperationAborted,
    operationCancelled                    OperationCancelled,
    serviceNotNegotiated                  ServiceNotNegotiated,
    transactionRolledBack                 TransactionRolledBack,
    specificExecuteDBLError               SpecificExecuteDBLError
  }

R-DefineDBL-Request          ::= SEQUENCE
  { commandHandle                     [0] CommandHandle,
    dataResourceHandle                [1] DataResourceHandle OPTIONAL,
    specificDBLStatement              [2] SpecificDBLStatement,
    specificDBLArgumentSpecification  [3] SpecificDBLArgumentSpecification OPTIONAL,
    specificDBLResultSpecification    [4] SpecificDBLResultSpecification OPTIONAL
  }

R-DefineDBL-Result           ::= SEQUENCE
  { specificDBLResultSpecification    [0] SpecificDBLResultSpecification OPTIONAL,
    specificDBLException              [1] SpecificDBLException OPTIONAL
  }

R-DefineDBL-Error            ::= CHOICE
  { dataResourceHandleNotSpecified        DataResourceHandleNotSpecified,
    dataResourceHandleUnknown             DataResourceHandleUnknown,
    duplicateCommandHandle                DuplicateCommandHandle,
    duplicateOperationID                  DuplicateOperationID,
    invalidSequence                       InvalidSequence,
    noDataResourceAvailable               NoDataResourceAvailable,
    operationAborted                      OperationAborted,
    operationCancelled                    OperationCancelled,
    serviceNotNegotiated                  ServiceNotNegotiated,
    specificDefineDBLError                SpecificDefineDBLError
  }
```

```
R-InvokeDBL-Request          ::= SEQUENCE
  { commandHandle                    [0] CommandHandle,
    dblArguments                    CHOICE
    { singleArgument                 [1] SEQUENCE
      { repetitionCount                [0] INTEGER DEFAULT 1,
        specificDBLArgumentValues      [1] SpecificDBLArgumentValues OPTIONAL
      },
      multipleArgument               [2]   SEQUENCE
      { listOfSpecificDBLArgumentValues [0]   SEQUENCE OF SpecificDBLArgumentValues
      }
    } OPTIONAL
  }

R-InvokeDBL-Result           ::= SEQUENCE
  { specificDBLResultSpecification    [0] SpecificDBLResultSpecification OPTIONAL,
    specificTerminationCode           [1] SpecificTerminationCode OPTIONAL,
    listOfResultValues                [2] SEQUENCE OF ResultValues OPTIONAL
  }

R-InvokeDBL-Error            ::= CHOICE
  { badRepetitionCount                BadRepetitionCount,
    commandHandleUnknown              CommandHandleUnknown,
    duplicateOperationID              DuplicateOperationID,
    invalidSequence                   InvalidSequence,
    operationAborted                  OperationAborted,
    operationCancelled                OperationCancelled,
    serviceNotNegotiated              ServiceNotNegotiated,
    transactionRolledBack             TransactionRolledBack,
    specificInvokeDBLError            SpecificInvokeDBLError
  }

R-DropDBL-Request            ::= SEQUENCE
  { listOfCommandHandle               [0] SEQUENCE OF CommandHandle OPTIONAL,
    specificDropDBLArgument          [30] SpecificDropDBLArgument OPTIONAL
  }

R-DropDBL-Result             ::= SEQUENCE
  { listOfDropDBLExceptions           [0] SEQUENCE OF DropDBLException OPTIONAL,
    specificDropDBLResult            [30] SpecificDropDBLResult OPTIONAL
  }

R-DropDBL-Error              ::= CHOICE
  { duplicateOperationID              DuplicateOperationID,
    invalidSequence                   InvalidSequence,
    operationAborted                  OperationAborted,
    operationCancelled                OperationCancelled,
    serviceNotNegotiated              ServiceNotNegotiated,
    specificDropDBLError              SpecificDropDBLError
  }
```

```
--Definitions of common types, ordered alphabetically:

AuthenticationData          ::= CHOICE
  { cstring                         [0] IA5String,
    ostring                         [1] OCTET STRING,
    bstring                         [2] BIT STRING
  }

CloseException              ::=  SEQUENCE
  { dataResourceHandle              [0] DataResourceHandle,
    closeException                 CHOICE
    { dataResourceHandleUnknown     [1] NULL,
      specificCloseException       [30] SpecificCloseException
    }
  }

CommandHandle               ::= INTEGER

DataResourceHandle          ::= INTEGER

DialogueIDClientInvocation  ::= SEQUENCE
  { apTitle                         [0] AP-title,
    aeQualifier                     [1] AE-qualifier,
    apInvocationID                  [2] AP-invocation-identifier,
    aeInvocationID                  [3] AE-invocation-identifier
  }

DialogueIDSuffix            ::= CHOICE
  { ostring                         [0] OCTET STRING
  }

DropDBLException            ::=  SEQUENCE
  { commandHandle                   [0] CommandHandle,
    dropDBLException               CHOICE
    { commandHandleUnknown          [1] NULL
      specificDropDBLException     [30] SpecificDropDBLException
    }
  }

FunctionalUnits             ::= BIT STRING
  {  termination                        (0),
     transaction                        (1),
     cancel                             (2),
     status                             (3),
     resource                           (4),
     immediate-DBL                      (5),
     stored-DBL                         (6)
   }
```

```
OperationID                 ::= INTEGER

ResultValues                ::=  SEQUENCE
  { specificDBLException           [0] SpecificDBLException OPTIONAL,
    specificDBLResultValues        [1] SpecificDBLResultValues OPTIONAL
  }

StatusInformation           ::=  SEQUENCE
  { operationID                    [0] OperationID,
    operationStatus               CHOICE
    { operationIDUnknown           [1] NULL,
      awaitingExecution            [2] NULL,
      executing                    [3] NULL,
      finished                     [4] NULL,
      cancelled                    [5] NULL,
      aborted                      [6] VisibleString
    },
    specificOperationStatusResult  [30] SpecificOperationStatusResult OPTIONAL
  }
```

```
--*********************************************************************
-- Definitions of generic ASN.1 errors:
 --*******************************************************************
-- Application tags for errors are assigned by the RDA Specialization.
-- In this template, these tags are shown as [APPLICATION nn]
-- where "nn" is a unique tag number for each Error
-- to be assigned by the RDA Specialization.
--*********************************************************************

AccessControlViolation          ::=  [APPLICATION  nn] NULL

BadRepetitionCount              ::=  [APPLICATION  nn] NULL

CommandHandleUnknown            ::=  [APPLICATION  nn] NULL

ControlAuthenticationFailure    ::=  [APPLICATION  nn] NULL

ControlServicesNotAllowed       ::=  [APPLICATION  nn] NULL

DataResourceAlreadyOpen         ::=  [APPLICATION  nn] SEQUENCE
  { dataResourceHandle                 [0] DataResourceHandle
  }

DataResourceHandleNotSpecified  ::=  [APPLICATION  nn] NULL

DataResourceHandleUnknown       ::=  [APPLICATION  nn] NULL

DataResourceNameNotSpecified    ::=  [APPLICATION  nn] NULL

DataResourceNotAvailable        ::=  [APPLICATION  nn] ErrorDiagnostic

DataResourceUnknown             ::=  [APPLICATION  nn] NULL

DialogueIDUnknown               ::=  [APPLICATION  nn] NULL

DuplicateCommandHandle          ::=  [APPLICATION  nn] NULL

DuplicateDataResourceHandle     ::=  [APPLICATION  nn] NULL

DuplicateDialogueID             ::=  [APPLICATION  nn] NULL

DuplicateOperationID            ::=  [APPLICATION  nn] NULL
```

```
InvalidSequence                 ::=  [APPLICATION  nn] SEQUENCE
  { diagnosticInformation            [0] ENUMERATED
    { dialogueNotActive                  (1),
      dialogueInitializing               (2),
      dialogueAlreadyActive              (3),
      transactionNotOpen                 (4),
      transactionOpen                    (5),
      transactionTerminating             (6),
      dialogueTerminating                (7)
    } OPTIONAL
  }

NoDataResourceAvailable         ::=  [APPLICATION  nn] NULL

OperationAborted                ::=  [APPLICATION  nn] ErrorDiagnostic

OperationCancelled              ::=  [APPLICATION  nn] NULL

ParentDataResourceHandleUnknown ::=  [APPLICATION  nn] NULL

ServiceNotNegotiated            ::=  [APPLICATION  nn] NULL

TransactionRolledBack           ::=  [APPLICATION  nn] NULL

UserAuthenticationFailure       ::=  [APPLICATION  nn] NULL

ErrorDiagnostic                 ::=  SEQUENCE
  { errorType                        [0] ENUMERATED
    { transient                          (0),
      permanent                          (1)
    } DEFAULT transient,
    diagnosticInformation            [1] VisibleString OPTIONAL
  }
```

```
--*********************************************************************
-- Specialization-defined parameters:
--*********************************************************************
-- The definition of the following "specific" types is left to each
-- RDA Specialization.  An RDA Specialization may choose to replace the
-- prefix "specific" in type and value references with something more
-- meaningful.  If a parameter referencing one of these "specific"
-- types is declared OPTIONAL, then that parameter may be deleted if
-- it is not needed by the RDA Specialization, or the keyword OPTIONAL
-- may be deleted to make that parameter mandatory.
--*********************************************************************

--          SpecificAccessControlData,
--          SpecificCancelArgument,
--          SpecificCancelResult,
--          SpecificCloseArgument,
--          SpecificCloseException,
--          SpecificCloseResult,
--          SpecificDBLArgumentValues,
--          SpecificDBLArgumentSpecification,
--          SpecificDBLException,
--          SpecificDBLResultSpecification,
--          SpecificDBLResultValues,
--          SpecificDBLStatement,
--          SpecificDropDBLArgument,
--          SpecificDropDBLException,
--          SpecificDropDBLResult,
--          SpecificInitializeArgument,
--          SpecificInitializeResult,
--          SpecificOpenArgument,
--          SpecificOpenResult,
--          SpecificOperationStatusResult,
--          SpecificStatusArgument,
--          SpecificStatusResult,
--          SpecificTerminateArgument,
--          SpecificTerminateResult,
--          SpecificTerminationCode,
--          SpecificUsageMode
```

```
--*********************************************************************
-- Specialization-defined errors:
--*********************************************************************
-- The definition of the following "specific" errors is left to each
-- RDA Specialization.  An RDA Specialization may choose to replace the
-- prefix "specific" in type and value references with something more
-- meaningful.  The RDA Specialization shall define these errors in a
-- manner consistent with the generic ASN.1 errors defined above.  The
 -- RDA Specialization may choose to delete the "specific" errors that
-- are not needed.
--*********************************************************************

--          SpecificCancelError,
--          SpecificCloseError,
--          SpecificDefineDBLError,
--          SpecificDropDBLError,
--          SpecificExecuteDBLError,
--          SpecificInitializeError,
--          SpecificInvokeDBLError,
--          SpecificOpenError,
--          SpecificStatusError,
--          SpecificTerminateError

END  -- RDA Specialization ASN.1 Module Template
```

4.4 Conformance

This clause states conformance requirements that are common to all RDA Specializations and all RDA application-contexts.

A claim of conformance to the RDA Generic Standard (this part of ISO/IEC 9579) is asserted by claiming conformance to an RDA Specialization Standard that references it.

NOTE – An implementation that claims conformance to an RDA Specialization referencing this part of ISO/IEC 9579 shall also conform to the specific conformance requirements for that RDA Specialization.

For the purposes of this clause, the term "RDA client role" means the ability to initiate RDA requests, and the term "RDA server role" means the ability to respond to such RDA requests.

4.4.1 Static conformance

An implementation that claims conformance to an RDA Specialization referencing this part of ISO/IEC 9579 shall provide

a) the RDA client role, the RDA server role, or both roles;

b) the RDA Dialogue Initialization functional unit (3.1.1.1);

c) at least one of the RDA functional units specified in the RDA Control services group (3.1.3) or the Database Language services group (3.1.5);

d) the Resource Handling functional unit (3.1.4), whenever any of the functional units in the Database Language services group is provided; and

e) for that RDA Specialization, one or more of the RDA application-contexts specified in section 5, Application-contexts.

NOTE – Specific conformance requirements are stated in section 5 for each application-context provided.

4.4.2 Dynamic conformance

An implementation that claims conformance to an RDA Specialization referencing this part of ISO/IEC 9579 shall exhibit external behaviour consistent with having implemented

a) if it provides the server role, the server execution rules specified in clause 4.1, Server execution rules;

b) the RDA protocol machine specified in clause 4.2, RDA protocol machine; and

c) the abstract syntax for application-protocol-data-units specified in clause 4.3, Application-protocol-data-units.

NOTE – This applies to those elements of the RDA ASN.1 that are fully specified in this part of ISO/IEC 9579. Certain elements of the RDA ASN.1 are left to be specified in each RDA Specialization Standard (as specified in clause 6.1, RDA Specialization Standards). Each RDA Specialization shall specify those elements and state what its conformance requirements are for them.

Conformance to the server execution rules and RDA protocol machine specification shall only be tested according to their effect on RDA APDU sequencing and on RDA APDU content.

Section 5: Application-contexts

This section contains specifications of the application-contexts that include the RDA ASE. For each application-context, there is

- a description of its purpose and scope;
- a list of the ASEs composing the context;
- the rules to be enforced by the context's single association control function, including sequencing rules and state transition diagrams, the specification of the mapping of the context's application-protocol-data-units onto underlying OSI services, concatenation rules, and rules for the interaction of the protocol machines of the ASEs that compose the context;
- any rules for use of optional features of the ASEs that compose the context; and
- conformance rules that are specific to the context.

5.1 RDA Basic application-context

5.1.1 Application-context name

Defined by an RDA Specialization.

5.1.2 Purpose and scope

5.1.2.1 General description

The RDA Basic application-context includes the services of RDA and ACSE for RDA transactions that access data resources through a single remote open system. The RDA Service provides the facilities for dialogue management, transaction management, and database access. The ACSE provides the facilities for the establishment of an association and for the normal and abnormal release of an association.

The RDA Basic application-context places additional constraints on the use of ACSE services, as specified in 5.1.4, SACF rules, and 5.1.6, Use of optional features, below.

5.1.2.2 RDA dialogue failure

If there is a failure of the RDA service-provider, both the RDA client and the RDA server are notified of the failure of the underlying association. If the failure is local to the database server, then the database server is notified by some local action following its recovery. Such local recovery is outside the scope of this part of ISO/IEC 9579.

Following failure of the RDA dialogue, recovery actions by the RDA server are required as follows:

- If there is an RDA transaction open (that is, no R-Commit or R-Rollback indication has been received), then the RDA server shall roll back the changes made to data resources by that RDA transaction.
- If there is an RDA transaction terminating (that is, an R-Commit or R-Rollback indication has been received), then the RDA server may attempt to commit the RDA transaction (but only if an R-Commit indication was received) or may roll it back.

The RDA Service provides no facilities for the RDA client to directly determine the outcome of an RDA transaction whose completion occurs following an R-Commit or R-Rollback request and RDA dialogue failure.

5.1.3 Set of ASEs

The RDA Basic application-context is composed of

- ACSE, as specified by ISO 8649; and
- RDA, as specified by this part of ISO/IEC 9579.

5.1.4 SACF rules

The SACF rules specified in this subclause are in addition to rules already specified by the ASEs listed in 5.1.3, Set of ASEs, above.

5.1.4.1 Association establishment and release

5.1.4.1.1 A-ASSOCIATE

- Rule 4.1.1.1: An A-ASSOCIATE request shall only be issued by the RDA client.

5.1.4.1.2 A-RELEASE

- Rule 4.1.2.1: An A-RELEASE request shall not be issued by either the RDA client or the RDA server while an RDA dialogue is active on the association.

5.1.4.1.3 A-ABORT

- Rule 4.1.3.1: An RDA client that issues an A-ABORT request or that receives an A-ABORT indication for a particular association shall delete all state information for the RDA dialogue active on that association.
- Rule 4.1.3.2: An RDA server that issues an A-ABORT request or that receives an A-ABORT indication for a particular association shall cause the RDA dialogue failure actions as specified in 4.1.2.8, Failure of the RDA dialogue, to be performed.

5.1.4.1.4 A-P-ABORT

- Rule 4.1.4.1: An RDA client that receives an A-P-ABORT indication for a particular association shall delete all state information for the RDA dialogue active on that association.
- Rule 4.1.4.2: An RDA server that receives an A-P-ABORT indication for a particular association shall cause the RDA dialogue failure actions as specified in 4.1.2.8, Failure of the RDA dialogue, to be performed.

5.1.4.2 RDA dialogue initialization and termination

5.1.4.2.1 R-Initialize

- Rule 4.2.1.1: Upon receipt of an R-Initialize request, the RDAPM of the RDA client shall assign an association compatible with the RDA dialogue (according to the rules stated in 5.1.6.1, A-ASSOCIATE) and send an R-Initialize-RI APDU. Upon receipt of an R-Initialize-RI APDU on an association, the RDAPM of the RDA server shall bind that association with the RDA dialogue.
- Rule 4.2.1.2: Upon receipt of an R-Initialize-RI APDU, the RDAPM of the RDA server shall set the value of the dialogueIDClientInvocation parameter to the value of the identification of the RDA client application-entity-invocation (that is, to the Calling AP Title, Calling AE Qualifier, Calling AP Invocation-identifier, and Calling AE Invocation-identifier parameters) from the A-ASSOCIATE indication primitive that established the association being used by the current RDA dialogue.

  NOTE – The value of the dialogueIDSuffix parameter is obtained from the R-Initialize-RI APDU.

- Rule 4.2.1.3: Upon receipt of an R-Initialize error response, the RDAPM of the RDA server shall send an R-Initialize-RC APDU and relinquish use of the association. Upon receipt of an R-Initialize-RC APDU carrying an error response, the RDAPM of the RDA client shall relinquish use of the association.

5.1.4.2.2 R-Terminate

- Rule 4.2.2.1: Upon receipt of an R-Terminate result response, the RDAPM of the RDA server shall send an R-Terminate-RC APDU and relinquish use of the association. Upon receipt of an R-Terminate-RC carrying a result response, the RDAPM of the RDA client shall relinquish use of the association.

5.1.4.3 Mapping rules

5.1.4.3.1 ACSE APDUs

No additional rules are required.

5.1.4.3.2 RDA APDUs

All RDA APDUs map onto the User data parameter of the P-DATA request (see ISO 8822).

5.1.5 State transition diagrams

Figures 2 and 3 show the state transitions of the RDA Basic application-context.

NOTE – State transitions caused by RDA Service errors are not illustrated in these figures.

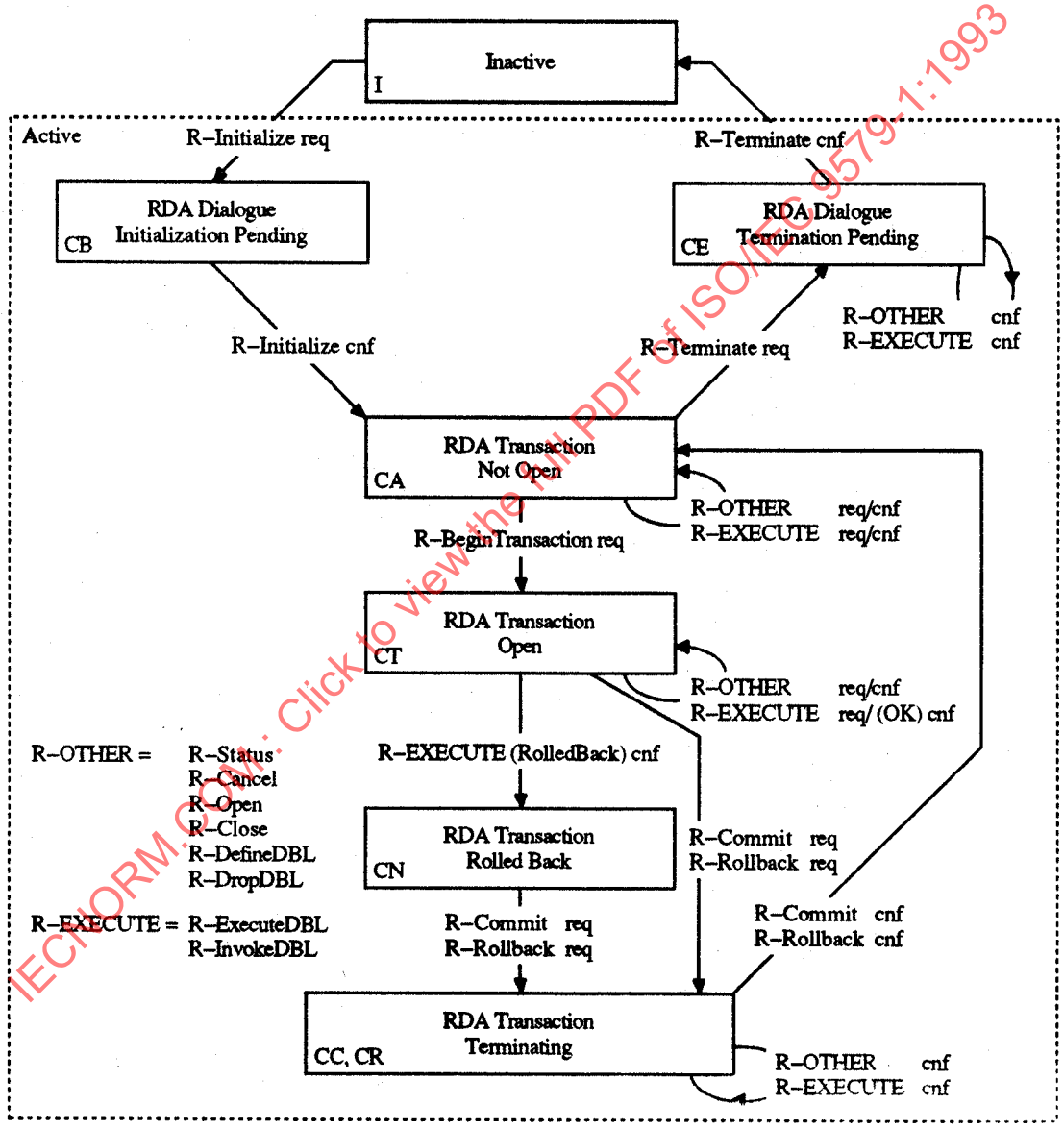

Figure 5 – State transition diagram for RDA Basic application-context – RDA client

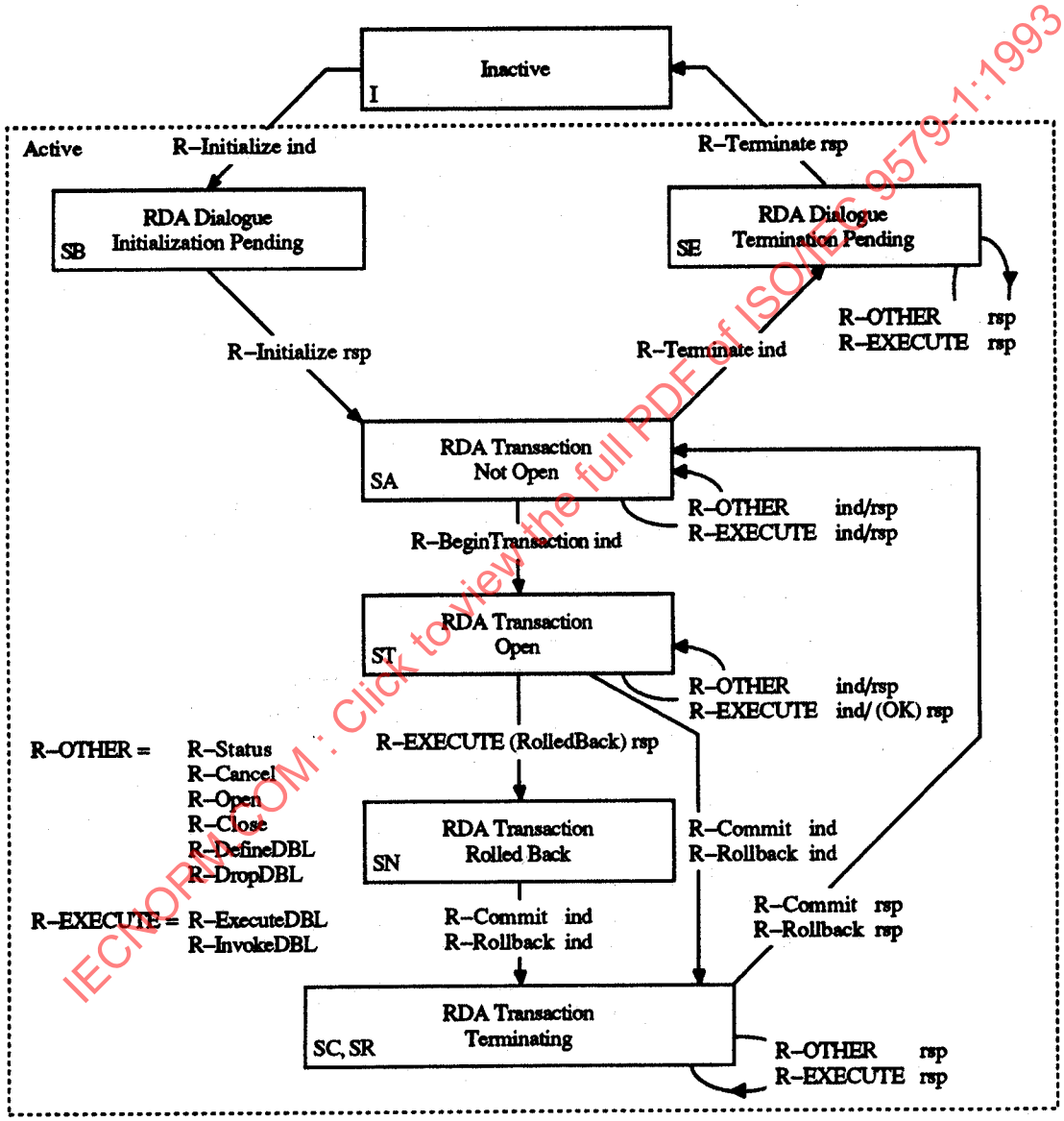

Figure 6 – State transition diagram for RDA Basic application-context – RDA server

5.1.6 Use of optional features

5.1.6.1 A-ASSOCIATE

- Rule 6.1.1: An A-ASSOCIATE request shall specify the application-context name of an RDA Basic application-context.
- Rule 6.1.2: An A-ASSOCIATE request shall specify the identification of the application-entity-invocation of the RDA client via the Calling AP Title, Calling AE Qualifier, Calling AP Invocation-Identifier, and Calling AE Invocation-Identifier parameters.
- Rule 6.1.3: An A-ASSOCIATE request shall identify the abtract syntaxes required for the RDA and ACSE ASEs via the Default Presentation Context Name and/or the Presentation Context Definition List parameters.

5.1.6.2 A-RELEASE

No additional rules are required.

5.1.6.3 A-ABORT

No additional rules are required.

5.1.6.4 A-P-ABORT

No additional rules are required.

5.1.7 Conformance

This subclause states additional conformance requirements that are in common to all RDA Specializations but that are specific to the RDA Basic application-context.

NOTE – These rules are in addition to those stated for every RDA application-context in clause 4.4, Conformance.

5.1.7.1 Static conformance

An implementation that claims conformance to an RDA Specialization referencing this part of ISO/IEC 9579 and that claims conformance to this application-context shall provide

a) the RDA Dialogue Termination functional unit (3.1.1.2);

b) the Presentation Kernel functional unit, in conformance with ISO 8823/Amd. 2 (to be published);

c) the Session Kernel and Duplex functional units of Version 2, in conformance with ISO 8327/Add. 2; and

d) the Association Control Service Element (ACSE), in conformance with ISO 8650.

5.1.7.2 Dynamic conformance

An implementation that claims conformance to an RDA Specialization referencing this part of ISO/IEC 9579 and that claims conformance to this application-context shall exhibit external behaviour consistent with having implemented

a) the single association control function rules specified in 5.1.4, SACF rules; and

b) the rules for the use of optional features specified in 5.1.6, Use of optional features.