X/Open Consortium Specification

SQL Remote Database Access
over OSI and Non-OSI Transport Providers

Published by X/Open on behalf of the SQL Access Group

A Consortium Specification is a document published by X/Open on behalf of an organisation. As such, it does not represent the results of any X/Open work programme. X/Open does not make any claims relating to quality, accuracy or industry consensus for such specifications.

This document is the result of a joint collaborative effort between the X/Open Data Management Working Group and the SQL Access Group's Formats and Protocols Working Group, and was developed before the SQL Access Group was absorbed into X/Open in December 1994. Publication of this specification is not intended to imply X/Open endorsement of its content.

X/Open Consortium Specification

SQL Remote Database Access over OSI and Non-OSI Transport Providers

Document Number: J501

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.org

Contents

List of Figures

List of Tables

Preface

The SQL Access Group, founded in 1989, is a consortium of major vendors and users of database software and hardware. It is committed to the promotion of worldwide, multi-vendor interoperability and portability of applications across distributed, SQL-based relational database systems. This goal is manifested in three primary activities:

- Development of specifications, both for application programming interfaces, based on the existing SQL standard (ISO SQL), and for communication protocols, based on the existing Remote Database Access standard (ISO RDA).
- Validation of these specifications through implementation of prototypes on multiple platforms.
- Promotion of these specifications, both by making them available to others for general industry use and by submitting them for consideration by standards-writing bodies, including the International Standards Organization (ISO) and the American National Standards Institute (ANSI).

The SQL Access Group bases its work on existing or emerging standards, and supplements these by specification of the additional details and agreements necessary to ensure database interoperability and portability.

This Document

This document specifies SQL Remote Database Access over OSI and Non-OSI Transport Providers.

This document is the result of a joint collaborative effort between the X/Open Data Management Working Group and the SQL Access Group's Formats and Protocols Working Group.

This document has not been formally adopted by X/Open as part of the X/Open Common Applications Environment (CAE). It has been published in order to allow industry review and comment, as requested by the SQL Access Group.

Structure

This document is structured as follows:

- Chapter 1 is a brief introduction to RDA and to the need for the refinements that X/Open has specified.
- Chapter 2 specifies the set of formal changes to the referenced ISO RDA documents that produces a version that is usable with X/Open SQL.
- Chapter 3 describes agreements on the use and maximum values of parameters for the X/Open RDA protocol.
- Chapter 4 specifies the requirements of X/Open RDA on the services provided by the OSI upper layers.
- Chapter 5 specifies how X/Open RDA permits the use of message transport facilities other than OSI transport facilities.

- Appendix A shows the resulting text of Table 10 of RDA SQL Specialization when the changes specified in Chapter 2 are applied.
- Appendix B shows the resulting ASN.1 module when the changes specified in Chapter 2 are applied.
- Appendix C specifies how X/Open RDA is provided on OSI transport providers.
- Appendix D specifies how X/Open RDA is provided on TCP transport providers.
- Appendix E contains X/Open's recommendations for application programming interfaces that can be used in implementations of X/Open RDA, and gives guidance on their use.
- A glossary and index are provided.

Revision History

This document differs from the X/Open **RDA** specification as follows:

- Referenced Documents has been extended to include the additional standards and specifications referenced in this version.
- Chapter 1 has been extended to explain the added chapters and appendices.
- In Chapter 3 the former Section 3.5, Protocol Stack has been deleted (it is replaced by Chapter 4).
- Chapter 4 has been added.
- Chapter 5 has been added.
- Appendix C has been added.
- Appendix D has been added.
- Appendix E has been added.

Typographical Conventions

The following typographical conventions are used throughout this document:

- **Bold** font is used in text for filenames, keywords, type names, data structures and their members.
- *Italic* strings are used for emphasis or to identify the first instance of a word requiring definition. Italics in text also denote variables or substitutable items.
- Normal font is used for the names of constants and literals.
- Syntax and code examples are shown in `fixed width` font.
- Variables within syntax statements are shown in *`italic fixed width`* font.

Because this specification defines changes to an ISO standard, the ISO typographical conventions are used in Chapter 2 and Appendix A. Refer to Section 2.1 on page 3 for details.

Trade Marks

UNIX® is a registered trade mark in the United States and other countries, licensed exclusively through X/Open Company Limited.

X/Open® is a registered trade mark, and the ‘‘X’’ device is a trade mark, of X/Open Company Limited.

Referenced Documents

The following standards are referenced in this specification:

ISO 7498
: ISO 7498: 1984, Information Processing Systems — Open Systems Interconnection — Basic Reference Model.

ISO 8072
: ISO 8072: 1986, Information Processing Systems — Open Systems Interconnection — Transport Service Definition.

ISO 8326
: ISO 8326: 1987, Information Processing Systems — Open Systems Interconnection — Basic Connection-oriented Session Service Definition.

ISO 8327
: ISO 8327: 1987, Information Processing Systems — Open Systems Interconnection — Basic Connection-oriented Session Protocol Specification (including AD2: 1988, Addendum 2: Incorporation of Unlimited User Data).

ISO 8650
: ISO 8650: 1992, Information Processing Systems — Open Systems Interconnection — Protocol Specification for the Association Control Service Element.

ISO 8823
: ISO 8823: 1988, Information Processing Systems — Open Systems Interconnection — Connection-oriented Presentation Protocol Specification.

ISO 8859-1
: ISO 8859-1: 1987, Information Processing — 8-bit Single-byte Coded Graphic Character Sets — Part 1: Latin Alphabet No. 1.

ISO/IEC 8073
: ISO/IEC 8073: 1987, Information Technology — Telecommunications and Information Exchange Between Systems — Open Systems Interconnection — Protocol for Providing the Connection-mode Transport Service.

Minimal OSI
: ISO/IEC DISP 11188-3, International Standardized Profile — Common Upper Layer Requirements — Part 3: Minimal OSI Upper Layers Facilities, Version 6, 1994-04-14.

OIW RDA
: Stable Implementation Agreements for Open Systems Interconnection Protocols: Part 19 — Remote Database Access, Output from the December 1992 OSE Implementors' Workshop.

RDA Generic
: ISO/IEC 9579-1: 1993, Information Technology — Open Systems Interconnection — Remote Database Access — Part 1: Generic Model, Service, and Protocol.

RDA ISP, Part 3
: ISO/IEC DISP xxxxx-3, Information Technology — Open Systems Interconnection —

International Standardized Profile RDA — Part 3: Support of Session, Presentation and ACSE Protocols (Source: European Workshop on Open Systems, October 1993).

RDA SQL Specialization
: ISO/IEC 9579-2: 1993, Information Technology — Open Systems Interconnection — Remote Database Access — Part 2: SQL Specialization.

RFC 1006
: ISO Transport Service on Top of the TCP, Version 3, May 1987, Marshall T. Rose and Dwight E. Cass, Network Working Group, Northrop Research and Technology Center.

SQL-92
: ISO/IEC 9075: 1992, Information Technology — Database Language SQL (technically identical to ANSI standard X3.135-1992).

TCP
: Transmission Control Protocol, RFC 793 (Defense Communication Agency, DDN Protocol Handbook, Volume II, DARPA Internet Protocols, (December 1985).

The following X/Open documents are referenced in this specification:

RDA
: X/Open CAE Specification, August 1993, Data Management: SQL Remote Database Access (ISBN: 1-872630-98-7, C307).

SQL
: X/Open CAE Specification, August 1992, Structured Query Language (SQL) (ISBN: 1-872630-58-8, C201).

XAP
: X/Open CAE Specification, September 1993, ACSE/Presentation Services API (XAP) (ISBN: 1-872630-91-X, C303).

XTI, Version 2
: X/Open CAE Specification, September 1993, X/Open Transport Interface (XTI), Version 2 (ISBN: 1-872630-97-9, C318).

mOSI Functionality
: X/Open Preliminary Specification, September 1993, Appendix H, Minimum OSI Functionality of the X/Open CAE Specification, September 1993, X/Open Transport Interface (XTI), Version 2 (ISBN: 1-872630-97-9, C318).

The following documents are useful as supplementary reading but are not directly referenced:

ASN.1
: ISO 8824: 1990 Information Technology — Open Systems Interconnection — Specification of Abstract Syntax Notation One (ASN.1).

IP
: Internet Protocol, RFC 791, September 1981.

ISO 8649
: ISO 8649: 1988, Information Processing Systems — Open Systems Interconnection — Service Definition for the Association Control Service Element.

ISO 8822
: ISO 8822: 1988, Information Processing Systems — Open Systems Interconnection — Connection-oriented Presentation Service Definition.

Chapter 1

Introduction

X/Open Remote Database Access (RDA) provides access to one or more SQL databases on a remote system. This document specifies X/Open RDA in terms of additions to the ISO Open Systems Interconnection (OSI) standards that define ISO RDA.

X/Open RDA is an application-service that is consistent with OSI. That is, it fits into the OSI Model (see ISO 7498) as an Application Layer service and makes use of the OSI upper-layer services as prescribed by the ISO standards for those services. However, X/Open RDA is both a subset and a superset of ISO RDA, and thus is not an ISO-standard OSI application.

The ISO/IEC RDA Generic standard defines a generic model, service and protocol for RDA. RDA SQL Specialization refines RDA Generic for use with SQL. (In this document a reference to ‘‘RDA standards’’ means both RDA Generic and RDA SQL Specialization.) These RDA standards specify a message format for communication of SQL database language statements to a remote database and define how SQL statements are supported by the RDA protocol. X/Open endorses these standards and incorporates relevant subsets of them by reference. Refer to these standards for definitions of terms and for additional supporting material not repeated here.

Various standardisation and implementors' groups have published additional specifications and implementation agreements that are needed to ensure interoperability. The additions specified by the SQL Access Group are described in this document. The SQL Access Group expects that ISO and other standardisation groups will incorporate this or comparable material into formal standards and implementation agreements.

This document defines the following additions to ISO RDA:

- Additions to the ISO RDA standards.

  These additions are required to support the features of SQL-92 beyond Entry Level that are included in the X/Open **SQL** specification. These additions are specified in Chapter 2 on page 3, Appendix A on page 45 and Appendix B on page 49.

- Implementation agreements for the X/Open RDA protocol.

  These agreements specify limits on the values and use of the parameters of the RDA protocol. These agreements are specified in Chapter 3 on page 13.

- Requirements on and implementation agreements for X/Open RDA's use of the OSI layers below RDA.

  These requirements and agreements are specified in Chapter 4 on page 35.

  Section 4.1 on page 36 specifies the requirements of X/Open RDA on the services provided by the OSI upper layers in terms of the Minimal OSI International Standardized Profile for the OSI upper layers. Since RDA's use of the OSI upper layers can be described in terms of mOSI, X/Open RDA can be supported by an implementation of a subset of the full capabilities of the OSI upper layers.

  The implementation agreements supplement the requirements; they are specified in Section 4.2 on page 38.

- Specification of rules that permit X/Open RDA clients and servers to interoperate using other types of message transport providers than those specified by the OSI lower-layer standards (the Transport Layer and below).

  Firstly, the concepts supporting this capability are described. Secondly, the requirements imposed on such non-OSI transport providers are specified. These concepts and requirements are contained in Chapter 5 on page 39. Finally, the specific rules for each supported transport provider are stated. These rules are contained in Appendix C on page 69 and Appendix D on page 71.

  The SQL Access Group anticipates that the specifiers of other transport protocols will develop similar specifications for their transport providers. As such specifications are produced and accepted for X/Open RDA, corresponding additional appendices will be added to this document.

- Recommendations for and guidance on the use of application programming interfaces (APIs) that can be employed in implementations of X/Open RDA.

  These APIs are to the OSI upper-layer services and to the various message transport providers. These recommendations and guidelines are contained in Appendix E on page 73.

Chapter 2

Changes to the ISO RDA Standards

This chapter specifies changes to RDA Generic and RDA SQL Specialization that are required to support the features of SQL-92 beyond Entry Level included in the X/Open **SQL** specification.

Applying the changes in this chapter to the referenced ISO documents produces a conceptual document called ‘‘X/Open RDA’’.

Specific items in this section will be removed from future editions if they are eventually adopted by ISO for the RDA SQL-92 Addendum.

Some changes specified in this chapter affect Table 10 of RDA SQL Specialization. Appendix A of this document contains a revised Table 10 based on all these proposals.

Some changes specified in this chapter affect the ASN.1 module for RDA. Appendix B of this document contains a revised ASN.1 module based on all these proposals.

2.1 Typographical Equivalence

X/Open RDA implicitly refers to the X/Open **SQL** specification to define the SQL statement syntax allowed in an **R-ExecuteDBL** request. Table 10 in Clause 4.1.7.1.1 of RDA SQL Specialization uses a different typography from the X/Open **SQL** specification to refer to syntactic elements of SQL. This chapter follows the ISO typography, with some additions. That is:

- A syntactic element in angle brackets in RDA SQL Specialization is the same as the corresponding syntactic element in italics in the X/Open **SQL** specification. The use of punctuation in the respective documents is not significant. For example, <update statement: positioned> in RDA SQL Specialization is the same as *update-statement-positioned* in the X/Open **SQL** specification.
- The syntactic elements <table definition> and <view definition> in RDA SQL Specialization correspond respectively to *create-table-statement* and *create-view-statement* in the X/Open **SQL** specification.
- The syntactic elements <create index statement> and <drop index statement> are introduced by this chapter and correspond respectively to *create-index-statement* and *drop-index-statement* in the X/Open **SQL** specification.

2.2 sQLDBLStatementNotAllowed Error

In the current RDA SQL Specialization, only transaction management statements cause an error if they are transmitted to the server. To allow for additional statements that also cause this error, apply the following changes to RDA SQL Specialization:

1. Replace all occurrences of sQLDBLTransactionStatementNotAllowed with sQLDBLStatementNotAllowed.
2. Replace all occurrences of SQLDBLTransactionStatementNotAllowed with SQLDBLStatementNotAllowed.
3. In Clause 3.1.5.1.1, R-ExecuteDBLService, under Error Parameters, replace the description of sQLDBLTransactionStatementNotAllowed with the following:

   The content of the sQLDBLStatement is one of the following statements not permitted by the RDA SQL Specialization:
   <allocate descriptor statement>
   <commit statement>
   <connect statement>
   <deallocate descriptor statement>
   <disconnect statement>
   <get descriptor statement>
   <get diagnostics statement>
   <rollback statement>
   <set connection statement>
   <set descriptor statement>
4. In Clause 4.1.7.1.1:

   Delete the second sentence of Note 1 in Table 10 (R-ExecuteDBL use of SQL argument and result parameters).

   Under Error Rules, replace the Predicate for sQLDBLTransactionStatementNotAllowed with:

   The RDA client requested the execution of an RDA SQL statement that is one of the SQL statements listed in Table 10 (R-ExecuteDBL use of SQL argument and result parameters) with Note 1.

2.3 SQL Diagnostics Information

Apply the following changes to RDA SQL Specialization:

1. In Clause 1.3.1 (terms defined in ISO SQL), add:

   <get diagnostics statement>

2. In Clause 3.1.4.1.1 (R-Open):

   Add the following as a third subparameter of sQLOpenArgument in the R-Open SQL Specific Service Parameters table:

	Req	Ind	Rsp	Cnf
sQLDiagnosticsRequested	U	C(=)		

   Add the following description after sQLConformanceLevel in the Request Parameters section:

   sQLDiagnosticsRequested:
   This parameter specifies the level of diagnostics information requested by the RDA client. If ‘‘always’’ is chosen, then the RDA client desires that SQLDiagnostics always be included in the results of subsequent R-ExecuteDBL requests. If ‘‘onRequest’’ is chosen, then the RDA client desires that SQLDiagnostics be included in the results of a subsequent R-ExecuteDBL request only when specifically requested on that individual R-ExecuteDBL request. If ‘‘never’’ is chosen, then the RDA client does not desire that SQLDiagnostics be included in the results of subsequent R-ExecuteDBL requests. The default value for sQLDiagnosticsRequested is ‘‘never’’.

3. In Clause 3.1.5.1.1 (R-ExecuteDBL):

   Add the following request parameter to the end of the list of request parameters in the R-ExecuteDBL SQL Specific Service Parameters table:

	Req	Ind	Rsp	Cnf
returnSQLDiagnostics	U	C(=)		

   Add the following description after listOfSQLDBLArgumentValues in the Request Parameters section:

   returnSQLDiagnostics:
   This parameter specifies the level of diagnostics information requested by the RDA client when ‘‘onRequest’’ was chosen by the RDA client for the sQLDiagnosticsRequested parameter on R-Open. If ‘‘true’’ is chosen, then the RDA client desires that SQLDiagnostics be included in the results of the request. If ‘‘false’’ is chosen, then the RDA client desires that SQLDiagnostics not be included in the results of the request. The default value for returnSQLDiagnostics is ‘‘false’’.

   Add the following subparameter after the sQLErrorText subparameter of the sQLDBLException result parameter in the R-ExecuteDBL SQL Specific Service Parameters table:

	Req	Ind	Rsp	Cnf
sQLDiagnostics			U	C(=)

Add the following description after sQLErrorText in the Result Parameters section:

sQLDiagnostics:
The semantics of the components of sQLDiagnostics are specified in Clause 18 (Diagnostics Management) of SQL-92. An optional parameter may be omitted if its value is null.

4. Add the following attribute to Clause 4.1.1.2 (Opened Data Resource Entity):

sQLDiagnosticsRequested:
This attribute specifies the level of diagnostics information requested by the RDA client.

‘‘always’’:
The RDA client desires that SQLDiagnostics always be included in the results of subsequent R-ExecuteDBL requests.

‘‘onRequest’’:
The RDA client desires that SQLDiagnostics be included in the results of a subsequent R-ExecuteDBL request only when specifically requested on that individual R-ExecuteDBL request.

‘‘never’’:
The RDA client does not desire that SQLDiagnostics be included in the results of subsequent R-ExecuteDBL request.

5. Add the following to the entity manipulation rules in Clause 4.1.6.1.1 (R-Open Service):

Attribute	Initial Value
sQLDiagnosticsRequested	The sQLDiagnosticsRequested parameter value on the R-Open indication primitive, if provided. Otherwise, this attribute contains the default value ‘‘never’’.

6. Add the following to the result rule for sQLDBLException in Clause 4.1.7.1.1 (R-ExecuteDBL Service):

If the sQLDiagnosticsRequested attribute of the opened data resource entity has a value of ‘‘always’’, then SQLDiagnostics shall be returned by the RDA server. If the sQLDiagnosticsRequested attribute of the opened data resource entity has a value of ‘‘onRequest’’ and the returnSQLDiagnostics parameter of the R-ExecuteDBL indication has a value of ‘‘true’’, then SQLDiagnostics shall be returned by the RDA server. Otherwise, sQLDiagnostics shall not be returned.

7. Add the following row to Table 10 (R-ExecuteDBL use of SQL argument and result parameters):

RDA SQL Statement To Be Executed	ArgSpec	ArgVal	ResSpec	ResVal
<get diagnostics statement>[1]				

8. In Clause 4.2.2 (ASN.1 Module):

   Replace SQLOpenArgument with the following:

```
SQLOpenArgument                 ::= SEQUENCE
  { charSet                         [0] OBJECT IDENTIFIER OPTIONAL,
    sQLConformanceLevel             [1] OBJECT IDENTIFIER OPTIONAL,
    sQLDiagnosticsRequested         [2] ENUMERATED
    { always                        (0),
      onRequest                     (1),
      never                         (2)
      } DEFAULT never
    }
```

   Add the following parameter at the end of the SEQUENCE for R-ExecuteDBL-Request:

```
    returnSQLDiagnostics            [6] BOOLEAN DEFAULT TRUE
```

   (Add a comma after the OPTIONAL that concludes dblArguments.)

   Replace SQLDBLException with the following:

```
SQLDBLException                 ::= SEQUENCE
  { sQLSTATE                        [0] VisibleString OPTIONAL,
    sQLCODE                         [1] INTEGER OPTIONAL,
    sQLErrorText                    [2] VisibleString OPTIONAL,
    sQLDiagnostics                  [3] SQLDiagnostics OPTIONAL
    }

SQLDiagnostics                  ::= SEQUENCE
  { rowCount                        [0] INTEGER OPTIONAL,
    exceptionList                   [3] SEQUENCE OF ExceptionInfo
    }

ExceptionInfo                   ::= SEQUENCE
  { returnedSQLSTATE                [0] VisibleString,
    classOrigin                     [1] VisibleString,
    subclassOrigin                  [2] VisibleString,
    messageText                     [3] VisibleString OPTIONAL
    }
```

2.4 VARCHAR Data Type

Apply the following changes to RDA SQL Specialization:

1. Append the following entry to Table 8 (sQLDBLArgumentSpecification and sQLDBLResultSpecification):

	Req	Ind	Rsp	Cnf
varcharType	S	S(=)	S	S(=)
charSet	U	C(=)	U	C(=)
length	M	M(=)	M	M(=)

2. Append the following to Clause 3.1.6.2 (sQLDBLArgumentSpecification and sQLDBLResultSpecification):

   varcharType:
   This parameter describes an item of varying length character data.

   charSet:
   This parameter uniquely identifies the specification of a coded character set. The character repertoire for character data associated with (described by) this parameter is the character repertoire specified in the identified coded character set specification. If the parameter is omitted, then the character set is the default established by the declaration during the execution of the R-Open service that opened the associated SQL database resource. If no default was established, then this parameter must be specified.

   length:
   This parameter specifies the maximum number of characters allowed for the corresponding varying length character data item.

3. Add the following entry after doublePrecisionItem in Table 9 (sQLDBLArgumentValues and sQLDBLResultValues):

	Req	Ind	Rsp	Cnf
varcharItem	S	S(=)	S	S(=)

4. Add the following after doublePrecisionItem description in Clause 3.1.6.3 (sQLDBLArgumentValues and sQLDBLResultValues):

   varcharItem:
   This parameter contains the value of a varying length character Data Variable. The encoding of the character data shall be the encoding specified in the coded character set specification identified by the corresponding charSet parameter.

5. In Clause 4.2.2 (ASN.1 Module), extend the typeDescriptor subparameter of the SQLDataTypeDescriptor parameter as follows:

```
varcharType                  [15] SEQUENCE
-- SQL type: varchar
{ charSet                         OBJECT IDENTIFIER OPTIONAL,
  length                          INTEGER
  }
```

6. In Clause 4.2.2 (ASN.1 Module), extend the dataItem subparameter of the SQLValue parameter as follows:

```
varcharItem                    [10] OCTET STRING
```

2.5 Dynamic SQL Statements

RDA SQL Specialization does not include support for the dynamic SQL statements provided by the X/Open **SQL** specification. To include support for these statements, apply the changes below to Table 10 (R-ExecuteDBL use of SQL argument and result parameters) of RDA SQL Specialization.

Add the following rows:

RDA SQL Statement To Be Executed	ArgSpec	ArgVal	ResSpec	ResVal
<allocate descriptor statement>[1]				
<deallocate descriptor statement>[1]				
<describe statement>	C [11]		C ← S [7]	
<dynamic close statement>				
<dynamic declare cursor>				
<dynamic delete statement: positioned>				
<dynamic fetch statement>			C → S [8,10]	C ← S
<dynamic open statement>	C → S (H) [10]	C → S (H)	C ← S [9]	
<dynamic update statement: positioned>	C → S (H)	C → S (H)		
<execute statement>	C → S (H) [10]	C → S (H)		
<execute immediate statement>	C → S	C → S		
<get descriptor statement>[1]				
<prepare statement>	C → S	C → S		
<set descriptor statement>[1]				

In X/Open RDA, SQL descriptor statements are processed locally at the RDA client. The RDA client does not send to the server any of the four SQL statements: <allocate descriptor statement>, <deallocate descriptor statement>, <get descriptor statement> or <set descriptor statement>.

If a <dynamic fetch statement>, <dynamic open statement> or <execute statement> refers to an SQL descriptor area, the RDA client sends the SQLDataTypeDescriptors corresponding to the descriptor items of the referenced descriptor area. If a <describe statement> refers to the name of an SQL descriptor using an <embedded variable name>, the RDA client does not send the contents of that host variable.

The notes for the dynamic SQL statements added to Table 10 of RDA SQL Specialization, as shown in Appendix A, specify this behaviour.

2.6 Connection Management Statements

RDA SQL Specialization does not include support for the SQL connection management statements provided by the X/Open **SQL** specification. To include support for these statements, modify RDA SQL Specialization as follows:

1. Add the following rows to Table 10 (R-ExecuteDBL use of SQL argument and result parameters):

RDA SQL Statement To Be Executed	ArgSpec	ArgVal	ResSpec	ResVal
<connect statement>[1] <disconnect statement>[1] <set connection statement>[1]				

2. In Clause 3.1.5.1.1, in the description of the sQLDBLStatementNotAllowed error (as modified by Section 2.2), insert the following statements in alphabetic order within the list:

   <connect statement>
   <disconnect statement>
   <set connection statement>

2.7 Data Definition Statements

RDA SQL Specialization does not include support for all the SQL data definition statements provided by the X/Open **SQL** specification. There is also one SQL data definition statement in RDA SQL Specialization that is not provided by the X/Open **SQL** specification. To accommodate these differences, apply the following changes to Table 10 (R-ExecuteDBL use of SQL argument and result parameters) of RDA SQL Specialization:

1. Delete the following row:

   <schema definition>

2. Add the following rows:

RDA SQL Statement To Be Executed	ArgSpec	ArgVal	ResSpec	ResVal
<alter table statement> <create index statement> <drop index statement> <drop table statement> <drop view statement> <revoke statement>				

Chapter 3

Implementation Agreements

This chapter describes implementation agreements at the protocol level on maximum values and on the use of parameters required with X/Open RDA.

3.1 Character Set

Clients and servers must support at least the characters from columns 2 to 7 inclusive of the character code chart in the Latin Alphabet No. 1 (see ISO 8859-1). To specify the character set of the Latin Alphabet No. 1 (see ISO 8859-1) (in the charSet parameter of the R-Open service), clients and servers use the object identifier specified in Section 3.7.2 on page 32.

3.2 Operation Limits

An X/Open-compliant server can process a minimum of 32 pending (outstanding) RDA operations on a single RDA dialogue. It may reject an RDA operation for an RDA dialogue if it already has 32 operations pending on that dialogue.

3.3 Transactions

The number of data definition statements that a server can execute within a transaction is implementation-defined but is at least one. For maximum interoperability, clients should send only one data definition statement within a transaction.

3.4 RDA Features Not Used in This Specification

This specification does not use the following features of RDA:

- the RDA operations **R-DefineDBL**, **R-InvokeDBL** and **R-DropDBL**
- control services on another dialogue
- RDA SQL TP Application Context.

Future X/Open SQL RDA Specifications may make use of some or all of these features of RDA.

3.5 RDA Parameters

This section describes how X/Open SQL RDA clients and servers use RDA parameters. Tables describe each parameter's usage and any limits that apply. The tables are presented in the same order as in the RDA standards.

Limitations on Parameters

The **Parameter** column specifies the parameter by name. The dagger (†) symbol in the **Parameter** column indicates that the parameter is an X/Open extension to RDA SQL Specialization.

The **Limitation** column describes any limitation of the RI/RC APDU parameter value in addition to the limits imposed by the RDA standards. Where the limitation includes a maximum value, all X/Open-compliant servers allow at least that maximum. It is implementation-defined whether the server supports a larger maximum value. Ranges defined by text such as ''from 1 to 32'' are always inclusive. To ensure interoperability, clients should restrict parameter values to the range shown in the tables. If a parameter value is outside this range, X/Open does not specify the resulting behaviour. The limit of the length of an RDA APDU applies even when the limits for parameters within the same RDA APDU add up to a greater value.

The * symbol in the **Limitations** column indicates that the source of a limitation is the **Limits** section in the X/Open **SQL** specification. This section describes limitations of SQL implementations to which an application should adhere for portability.

Subparameters

Each RDA parameter appears on a separate line. Some lines describe subparameters (structures of component parameters). The structure is shown by the bullet (•) symbol in the **Parameter** column. A parameter name preceded by one or more bullets is a subparameter of the nearest entry above it that has one fewer bullet.

Mandatory Parameters

The ASN.1 module in Appendix B specifies whether a parameter is mandatory or optional. For those parameters that the RDA standards also define, in no case does the X/Open **SQL** specification differ from the RDA standards as to whether a parameter is mandatory or optional.

Unused Parameters

When the limitations below specify that a server **ignores** a parameter, it means that X/Open does not define the parameter's usage. The server must not reject a request based on the presence, absence or contents of the parameter.

Magnitude of INTEGER Parameters

Unless otherwise specified in the following tables, an INTEGER parameter is limited to a value from –2,147,483,648 to 2,147,483,647 (maximum 4 octets).

3.5.1 R-Initialize Service

Table 3-1 Parameters for R-Initialize request

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
dialogueID	See subparameters below.
•dialogueIDClientInvocation	See subparameters below.
••aP-title	No additional limitation.
••aE-qualifier	No additional limitation.
••aP-invocationID	No additional limitation.
••aE-invocationID	No additional limitation.
•dialogueIDSuffix	An OCTET STRING from 1 to 64 octets long.
identityOfUser	A VisibleString from 1 to 64 characters long.
userAuthenticationData	The client may provide an IA5String from 1 to 255 characters long, an OCTET STRING from 1 to 255 octets long, or a BIT STRING from 1 to 2040 bits long. If the server does not use this parameter, it ignores the parameter.
controlServiceDataRequested	A BOOLEAN whose value should be FALSE since X/Open-compliant servers are not required to !support control services on another dialogue.
functionalUnitsRequested	The client should not request the Stored Execution DBL functional unit. X/Open-compliant implementations are not required to support the Stored Execution DBL functional unit.
sQLInitializeArgument	See subparameters below.
•sQLConformanceLevelDefault	An OBJECT IDENTIFIER. See Section 3.7 on page 32.
•userData	An OCTET STRING from 1 to 255 octets long. X/Open does not specify its content.

Table 3-2 Parameters for R-Initialize result response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
controlServiceData	See subparameters below.
•controlServicesAllowed	No additional limitation.
•controlAuthenticationData	The server may provide an IA5String from 1 to 255 characters long, an OCTET STRING from 1 to 255 octets long, or a BIT STRING from 1 to 2040 bits long.
functionalUnitsAllowed	No additional limitation.
sQLInitializeResult	See subparameters below.
•userData	An OCTET STRING from 1 to 255 octets long. X/Open does not specify its content.

Table 3-3 Parameters for R-Initialize error response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
accessControlViolation	No additional limitation.
duplicateDialogueID	No additional limitation.
invalidSequence	See subparameter below.
•diagnosticInformation	No additional limitation.
operationAborted	See subparameters below.
•errorType	No additional limitation.
•diagnosticInformation	A VisibleString from 1 to 254 characters long.
userAuthenticationFailure	No additional limitation.

3.5.2 R-Synchronize APDU

The R-Synchronize-RI APDU has no parameters.

3.5.3 R-Terminate Service

Table 3-4 Parameters for R-Terminate request

Parameter	Limitation
`operationID`	An INTEGER with value greater than 0.

Table 3-5 Parameters for R-Terminate result response

Parameter	Limitation
`operationID`	An INTEGER with value greater than 0.

Table 3-6 Parameters for R-Terminate error response

Parameter	Limitation
`operationID`	An INTEGER with value greater than 0.
`duplicateOperationID`	No additional limitation.
`invalidSequence`	See subparameter below.
•`diagnosticInformation`	No additional limitation.
`operationAborted`	See subparameters below.
•`errorType`	No additional limitation.
•`diagnosticInformation`	A VisibleString from 1 to 254 characters long.
`serviceNotNegotiated`	No additional limitation.

3.5.4 R-BeginTransaction Service

Table 3-7 Parameters for R-BeginTransaction request

Parameter	Limitation
operationID	An INTEGER with value greater than 0.

Table 3-8 Parameters for R-BeginTransaction error response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
duplicateOperationID	No additional limitation.
invalidSequence	See subparameter below.
•diagnosticInformation	No additional limitation.
operationAborted	See subparameters below.
•errorType	No additional limitation.
•diagnosticInformation	A VisibleString from 1 to 254 characters long.
serviceNotNegotiated	No additional limitation.

3.5.5 R-Commit Service

Table 3-9 Parameters for R-Commit request

Parameter	Limitation
`operationID`	An INTEGER with value greater than 0.

Table 3-10 Parameters for R-Commit result response

Parameter	Limitation
`operationID`	An INTEGER with value greater than 0.
`transactionResult`	No additional limitation.

Table 3-11 Parameters for R-Commit error response

Parameter	Limitation
`operationID`	An INTEGER with value greater than 0.
`duplicateOperationID`	No additional limitation.
`invalidSequence`	See subparameter below.
•`diagnosticInformation`	No additional limitation.

3.5.6 R-Rollback Service

Table 3-12 Parameters for R-Rollback request

Parameter	Limitation
`operationID`	An INTEGER with value greater than 0.

Table 3-13 Parameters for R-Rollback result response

Parameter	Limitation
`operationID`	An INTEGER with value greater than 0.

Table 3-14 Parameters for R-Rollback error response

Parameter	Limitation
`operationID`	An INTEGER with value greater than 0.
`duplicateOperationID`	No additional limitation.
`invalidSequence`	See subparameter below.
•`diagnosticInformation`	No additional limitation.

3.5.7 R-Cancel Service

Table 3-15 Parameters for R-Cancel request

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
controlledDialogue	This parameter and its subparameters should be omitted because X/Open-compliant servers are not required to support controlled dialogues.
•dialogueID	See controlledDialogue.
••dialogueIDClientInvocation	See controlledDialogue.
•••aP-title	See controlledDialogue.
•••aE-qualifier	See controlledDialogue.
•••aP-invocationID	See controlledDialogue.
•••aE-invocationID	See controlledDialogue.
••dialogueIDSuffix	See controlledDialogue.
•controlAuthenticationData	See controlledDialogue.
listOfOperationID	This list may contain from 1 to 32 elements.
•OperationID	An INTEGER with value greater than 0.

Table 3-16 Parameters for R-Cancel result response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.

Table 3-17 Parameters for R-Cancel error response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
controlAuthenticationFailure	No additional limitation.
controlServicesNotAllowed	No additional limitation.
dialogueIDUnknown	No additional limitation.
duplicateOperationID	No additional limitation.
invalidSequence	See subparameter below.
•diagnosticInformation	No additional limitation.
operationAborted	See subparameters below.
•errorType	No additional limitation.
•diagnosticInformation	A VisibleString from 1 to 254 characters long.
serviceNotNegotiated	No additional limitation.

3.5.8 R-Status Service

Table 3-18 Parameters for R-Status request

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
controlledDialogue	This parameter and its subparameters should be omitted because X/Open-compliant servers are not required to support controlled dialogues.
•dialogueID	See controlledDialogue.
••dialogueIDClientInvocation	See controlledDialogue.
•••aP-title	See controlledDialogue.
•••aE-qualifier	See controlledDialogue.
•••aP-invocationID	See controlledDialogue.
•••aE-invocationID	See controlledDialogue.
••dialogueIDSuffix	See controlledDialogue.
•controlAuthenticationData	See controlledDialogue.
listOfOperationID	This list may contain from 1 to 32 elements.
•OperationID	An INTEGER with value greater than 0.

Table 3-19 Parameters for R-Status result response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
listOfStatusInformation	This list may contain from 1 to 32 elements.
•StatusInformation	See subparameters below.
••operationID	An INTEGER with value greater than 0.
••operationStatus	See choices below.
•••operationIDUnknown	No additional limitation.
•••awaitingExecution	No additional limitation.
•••executing	No additional limitation.
•••finished	No additional limitation.
•••cancelled	No additional limitation.
•••aborted	A VisibleString from 1 to 254 characters long.

Table 3-20 Parameters for R-Status error response

Parameter	Limitation
`operationID`	An INTEGER with value greater than 0.
`controlAuthenticationFailure`	No additional limitation.
`controlServicesNotAllowed`	No additional limitation.
`dialogueIDUnknown`	No additional limitation.
`duplicateOperationID`	No additional limitation.
`invalidSequence`	See subparameter below.
•`diagnosticInformation`	No additional limitation.
`operationAborted`	See subparameters below.
•`errorType`	No additional limitation.
•`diagnosticInformation`	A VisibleString from 1 to 254 characters long.
`serviceNotNegotiated`	No additional limitation.

3.5.9 R-Open Service

Table 3-21 Parameters for R-Open request

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
dataResourceHandle	An INTEGER with value greater than 0.
dataResourceName	A VisibleString from 1 to 254 characters long.
sQLAccessControlData	The client may provide an IA5String from 1 to 255 characters long, an OCTET STRING from 1 to 255 octets long, or a BIT STRING from 1 to 2040 bits long. If the server does not use this parameter, it ignores the parameter.
sQLUsageMode	No additional limitation.
sQLOpenArgument	See subparameters below.
•charSet	An OBJECT IDENTIFIER from 2 to 16 elements long.
•sQLConformanceLevel	An OBJECT IDENTIFIER. See Section 3.7 on page 32.
•sQLDiagnosticsRequested †	No additional limitation.

Table 3-22 Parameters for R-Open result response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
sQLOpenResult	See subparameters below.
•charSet	An OBJECT IDENTIFIER from 2 to 16 elements long.
•charSetNotSupported	No additional limitation.
•sQLConformanceLevel	An OBJECT IDENTIFIER. See Section 3.7 on page 32.

Table 3-23 Parameters for R-Open error response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
dataResourceAlreadyOpen	See subparameter below.
•dataResourceHandle	An INTEGER with value greater than 0.
dataResourceNameNotSpecified	No additional limitation.
dataResourceNotAvailable	See subparameters below.
•errorType	No additional limitation.
•diagnosticInformation	A VisibleString from 1 to 254 characters long.
dataResourceUnknown	No additional limitation.
duplicateDataResourceHandle	No additional limitation.
duplicateOperationID	No additional limitation.
invalidSequence	See subparameter below.
•diagnosticInformation	No additional limitation.
operationAborted	See subparameters below.
•errorType	No additional limitation.
•diagnosticInformation	A VisibleString from 1 to 254 characters long.
operationCancelled	No additional limitation.
serviceNotNegotiated	No additional limitation.
sQLOpenError	See choices below.
•invalidSQLConformanceLevel	No additional limitation.
•rDATransactionOpen	No additional limitation.
•sQLAccessControlViolation	No additional limitation.
•sQLDatabaseResourceAlreadyOpen	No additional limitation.

3.5.10 R-Close Service

Table 3-24 Parameters for R-Close request

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
listOfDataResourceHandle	This list must contain only one element.
•DataResourceHandle	An INTEGER with value greater than 0.

Table 3-25 Parameters for R-Close result response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
listOfCloseExceptions	This list must contain only one element.
•CloseException	See subparameters below.
••dataResourceHandle	An INTEGER with value greater than 0.
••closeException	See choice below.
•••dataResourceHandleUnknown	No additional limitation.

Table 3-26 Parameters for R-Close error response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
duplicateOperationID	No additional limitation.
invalidSequence	See subparameter below.
•diagnosticInformation	No additional limitation.
operationAborted	See subparameters below.
•errorType	No additional limitation.
•diagnosticInformation	A VisibleString from 1 to 254 characters long.
operationCancelled	No additional limitation.
serviceNotNegotiated	No additional limitation.
sQLCloseError	See choice below.
•rDATransactionOpen	No additional limitation.

3.5.11 R-ExecuteDBL Service

Table 3-27 Parameters for R-ExecuteDBL request

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
dataResourceHandle	An INTEGER with value greater than 0.
sQLDBLStatement	See subparameters below.
•statementText	An OCTET STRING from 1 to 4000* octets long.
•charSet	An OBJECT IDENTIFIER from 2 to 16 elements long.
sQLDBLArgumentSpecification	This parameter may contain from 1‡ to 100 entries of SQLDataTypeDescriptor. See Table 3-30 on page 29.
sQLDBLResultSpecification	This parameter may contain from 1‡ to 100 entries of SQLDataTypeDescriptor. See Table 3-30 on page 29.
dBLArguments	See choices below.
•singleArgument	See subparameters below.
••repetitionCount	An INTEGER with value from 1 to 64.
••sQLDBLArgumentValues	This parameter may contain from 1‡ to 100 entries of sQLValue. See Table 3-32 on page 31.
•multipleArguments	See subparameters below.
••listOfSQLDBLArgumentValues	This list may contain from 1 to 64 elements.
•••SQLDBLArgumentValues	This parameter may contain from 1‡ to 100 entries of sQLValue. See Table 3-32 on page 31.
returnSQLDiagnostics †	No additional limitation.

Table 3-28 Parameters for R-ExecuteDBL result response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
sQLDBLResultSpecification	This parameter may contain from 1‡ to 100 entries of SQLDataTypeDescriptor. See Table 3-30 on page 29.
listOfResultValues	This list may contain from 1 to 64 elements.
•ResultValues	See subparameters below.
••sQLDBLException	See Table 3-31 on page 30.
••sQLDBLResultValues	This parameter may contain from 1‡ to 100 entries of sQLValue. See Table 3-32 on page 31.

‡ For maximum interoperability, the sender should not send this parameter with zero elements. However, the receiver should accept this parameter with zero elements and process it as though the sender omitted the parameter.

Table 3-29 Parameters for R-ExecuteDBL error response

Parameter	Limitation
operationID	An INTEGER with value greater than 0.
badRepetitionCount	No additional limitation.
dataResourceHandleNotSpecified	No additional limitation.
dataResourceHandleUnknown	No additional limitation.
duplicateOperationID	No additional limitation.
invalidSequence	See subparameter below.
•diagnosticInformation	No additional limitation.
noDataResourceAvailable	No additional limitation.
operationAborted	See subparameters below.
•errorType	No additional limitation.
•diagnosticInformation	A VisibleString from 1 to 254 characters long.
operationCancelled	No additional limitation.
serviceNotNegotiated	No additional limitation.
transactionRolledBack	No additional limitation.
sQLExecuteDBLError	See choices below.
•hostIdentifierError	No additional limitation.
•rDATransactionNotOpen	No additional limitation.
•sQLDBLArgumentCountMismatch	No additional limitation.
•sQLDBLArgumentTypeMismatch	No additional limitation.
•sQLDBLNoCharSet	No additional limitation.
•sQLDBLStatementNotAllowed †	No additional limitation.
•sQLUsageModeViolation	No additional limitation.

3.6 Limits for Common Parameters

This section describes the parameters of common RDA types.

Table 3-30 Parameters for SQLDataTypeDescriptor

Parameter	Limitation
`nullable`	No additional limitation.
`colName`	A VisibleString from 1 to 18* characters long.
`typeDescriptor`	See choices below.
•`characterType`	See subparameters below.
••`charSet`	An OBJECT IDENTIFIER from 2 to 16 elements long.
••`length`	An INTEGER representing the maximum number of total text characters in the data value. Trailing blanks are included in the length, but any trailing null byte is not. If this `sQLDataTypeDescriptor` defines the data type of an argument value that contains an `sQLDBLStatement`, the value does not exceed 4000*. Otherwise, the value is from 1 to 254*.
••`fixedLengthEncoding`	No additional limitation.
•`numericType`	See subparameters below.
••`precision`	This value must be from 1 to 15.
••`scale`	This value must be from 0 to the value of `precision`.
•`decimalType`	See subparameters below.
••`precision`	This value must be from 1 to 15.
••`scale`	This value must be from 0 to the value of `precision`.
•`integerType`	See subparameters below.
••`precision`	If `precisionBase` indicates that `precision` is a number of decimal digits, then `precision` must be in the range from 1 to 10*. If `precisionBase` indicates that `precision` is a number of binary digits, then `precision` must be in the range from 1 to 31*.
••`precisionBase`	No additional limitation.
•`smallIntType`	See subparameters below.
••`precision`	If `precisionBase` indicates that `precision` is a number of decimal digits, then `precision` must be in the range from 1 to 5*. If `precisionBase` indicates that `precision` is a number of binary digits, then `precision` must be in the range from 1 to 15*.
••`precisionBase`	No additional limitation.
•`floatType`	See subparameters below.
••`mantissaPrecision`	An INTEGER with value from 1 to 47*.
••`maxExponent`	An INTEGER with value from 0 to 38*.
•`realType`	See subparameters below.
••`mantissaPrecison`	An INTEGER with value from 1 to 21*.
••`maxExponent`	An INTEGER with value from 0 to 38*.
•`doublePrecisionType`	See subparameters below.
••`mantissaPrecison`	An INTEGER with value from 1 to 47*.

Parameter	Limitation
••maxExponent	An INTEGER with value from 0 to 38*.
•varCharType †	See subparameters below.
••charSet †	An OBJECT IDENTIFIER from 2 to 16 elements long.
••length †	An INTEGER representing the maximum number of total text characters in the data value. Trailing blanks are included in the length, but any trailing null byte is not. If this sQLDataTypeDescriptor defines the data type of an argument value that contains an sQLDBLStatement, the value does not exceed 4000*. Otherwise, the value is from 1 to 254*.

Table 3-31 Parameters for SQLDBLException

Parameter	Limitation
sQLState	A server returns this 5-character VisibleString on any sQLDBLException it returns from **R-ExecuteDBL**. For valid values, see the Appendix B of the X/Open **SQL** specification.
sQLCode	This parameter corresponds to the deprecated SQLCODE status variable. Servers may return sQLCode in addition to sQLState.
sQLErrorText	A server may return a VisibleString from 1 to 254* characters long on any sQLDBLException it returns from **R-ExecuteDBL**.
sQLDiagnostics †	See subparameters below.
•rowCount †	No additional limitation.
•exceptionList †	This list may contain from 1 to 10 entries of exceptionInfo.
••exceptionInfo †	See subparameters below.
•••returnedSQLState †	A 5-character VisibleString. For valid values, see Appendix B of the X/Open **SQL** specification.
•••classOrigin †	A VisibleString from 1 to 254* characters long. Valid values correspond to those for the CLASS_ORIGIN field of the diagnostics area. See Section 5.8.1 of the X/Open **SQL** specification.
•••subclassOrigin †	A VisibleString from 1 to 254* characters long. Valid values correspond to those for the SUBCLASS_ORIGIN field of the diagnostics area. See Section 5.8.1 of the X/Open **SQL** specification.
•••messageText †	A VisibleString from 1 to 254* characters long.

Table 3-32 Parameters for SQLValue

Parameter	Limitation
`dataItem`	See choices below.
•`characterItem`	An OCTET STRING from 1 to 254* characters long.
•`numericItem`	An INTEGER with absolute value less than 10^{15}*.
•`decimalItem`	An INTEGER with absolute value less than 10^{15}*.
•`integerItem`	An INTEGER with a value from –2,147,483,648* to 2,147,483,647*.
•`smallIntItem`	An INTEGER with a value from –-32,768* to 32,767*.
•`floatItem`	A REAL with a value of 0 or absolute value within the range of 10^{-38}* to 10^{38}*.
•`realItem`	A REAL with a value of 0 or absolute value within the range of 10^{-38}* to 10^{38}*.
•`doublePrecisionItem`	A REAL with a value of 0 or absolute value within the range of 10^{-38}* to 10^{38}*.
•`varcharItem` †	An OCTET STRING from 0 to 254* characters long.
`indicator`	No additional limitation.

3.7 Object Identifiers

This specification defines certain object identifiers for cases in which it is necessary to distinguish between RDA SQL Specialization and X/Open RDA. It also references object identifiers defined by other authorities.

3.7.1 Object Identifiers Defined By X/Open

All X/Open-defined object identifiers begin with the following elements:

```
{ iso (1) national-member-body (2) uk (826)
  national (0) x-open (1050) data-management (3) ...
```

The following object identifiers are used in this specification:

Usage	Object Identifier
sQLConformanceLevel (without optional integrity feature)	`... x-open-sql (1) year-1992 (1) full-conformance (3) integrity-no (0) }`
sQLConformanceLevel (with optional integrity feature)	`... x-open-sql (1) year-1992 (1) full-conformance (3) integrity-yes (1) }`
Application Context Name*	`... x-open-rdasql (2) basic-ac (2) version-1 (1) }`
RDA Abstract Syntax*	`... x-open-rdasql (2) abstract-syntax (1) version-1 (1) }`
ASN.1 Module*	`... x-open-rdasql (2) module (0) version-1 (1) }`

An X/Open-compliant implementation must support the X/Open-defined object identifiers for Application Context Name, RDA Abstract Syntax and ASN.1 Module. In addition, an X/Open-compliant implementation must indicate its level of functionality through the use of the sQLConformanceLevel object identifier.

3.7.2 Object Identifiers Defined By Other Authorities

The following object identifier defined by another authority is also used in this specification:

Usage	Object Identifier
charSet **	`{ iso (1) identified-organization (3) oiw (14) rda-sig (9) character-sets (1) oiw-latin-1 (1) abstract-syntax (1) }`

The meaning of this object identifier is defined in OIW RDA.

---

* When the features required to support X/Open SQL RDA are provided by the RDA standards, X/Open-compliant implementations will be expected to use the object identifiers from these standards.

** X/Open-compliant implementations use this object identifier to identify a character set. If ISO should select an object identifier to refer to this character set, X/Open-compliant implementations are expected to use it instead of the object identifier in this specification.

3.8 Prospective Uses of Parameters

The `repetitionCount` parameter in the **R-ExecuteDBL** request directs the server to repeat the execution of the SQL statement the specified number of times, using the same argument. A `repetitionCount` greater than 1 might be useful when executing:

- INSERT
- FETCH (both static and dynamic)
- EXECUTE (or EXECUTE IMMEDIATE).

(Currently, there is no occasion in the X/Open SQL application interface that would call for use of this feature.)

The `listOfSQLDBLArgumentValues` parameter in the **R-ExecuteDBL** request directs the server to repeat the execution of the SQL statement the specified number of times, using a successive argument from the list each time. This parameter might be useful when executing:

- INSERT
- Searched DELETE
- Searched UPDATE
- EXECUTE (if the statement to be executed is one of the above)
- SELECT.

(Currently, there is no occasion in the X/Open SQL application interface that would call for use of this feature.)

Chapter 4

Requirements on the OSI Upper Layers

This chapter specifies the requirements of X/Open RDA on the services provided by the OSI upper layers in terms of the minimal OSI (mOSI) profile (see Minimal OSI).

mOSI is a profile of the OSI upper layers: the Association Control Service Element, the Presentation Layer and the Session Layer. A profile defines a combination of base standards that collectively perform a specific, well-defined function. It standardises the use of options and other variations in the standards to which it applies, and thus provides a basis for interoperability. The mOSI profile specifies a subset of the services of the OSI upper layers. (See Section E.1.2 on page 74 for discussion of the motivation for mOSI.)

Since mOSI is a subset of the OSI upper layers, an implementation using a mOSI implementation of the OSI upper layers and an implementation using a full implementation of the OSI upper layers will interoperate. This, of course, applies to implementations that require only that subset of full OSI that mOSI provides.

The mOSI specification permits an application specification to claim compliance with the mOSI profile by reference to the mOSI profile instead of repeating its provisions. X/Open RDA complies with the mOSI profile. This is demonstrated as follows:

- The requirements of ISO RDA on the OSI upper layers are stated in Annex A, Session, Annex B, Presentation and Annex C, ACSE of RDA ISP, Part 3.
- The requirements of mOSI on the OSI upper layers are stated in the Annex C, Session, Annex B, Presentation, and Annex A, ACSE, of Minimal OSI.
- Comparison of the tables in the corresponding annexes shows that all the PDUs and parameters required by the ISO RDA profiles for the RDA Basic application-context (ARD-11x in RDA ISP, Part 3) are provided by the mOSI profile. Since ISO RDA's use of the OSI upper layers can be described in terms of the mOSI profile, ISO RDA can claim compliance to the mOSI profile.
- X/Open RDA includes only the RDA Basic application-context, and uses only features of the OSI upper layers that are provided in the ISO RDA Basic application-context profiles. Since X/Open RDA's use of the OSI upper layers can be described in terms of the mOSI profile, X/Open RDA can also claim compliance to the mOSI profile.

4.1 mOSI Profile Requirements

A mOSI-compliant application specification must provide the information specified in Clause D.1 of Minimal OSI. This section provides that information for X/Open RDA.

4.1.1 Profile Requirements List Proforma

Table 4-1 completes Table D.1, Profile Requirements List Proforma, of Minimal OSI. Since RDA defines two roles, *client* and *server*, under the heading ''Specification's choice'' the table shows two columns for the mOSI variables.

The terms and conventions used in Table 4-1 and Table 4-2 on page 37 are defined in Clause 6, Conventions, of Minimal OSI:

m Mandatory Support.

i Out of Scope.

o Optional Support.

Table 4-1 RDA Profile Requirements List

	Item/Variable	Specification's Choice Client	Server
1	Establishment-initiator	**m**	**i**
2	Establishment-responder	**i**	**m**
3	Establishment-responder-reject	**i**	**m**
4	Normal-data-requestor	**m**	**m**
5	Normal-data-acceptor	**m**	**m**
6	Release-requestor	**m**	**i**
7	Release-acceptor	**i**	**m**
8	Authentication	**i**	**i**
9	Application-context-negotiation	**i**	**i**
10	Transport-expedited	**o**	**o**
11	Number of presentation-contexts required	2	
12	ISO/IEC ISP 11188-1 compliance?	yes	
13	Status values for all open (*) parameters	mixed (see Table 4-2)	

4.1.2 Open Parameters

Table 4-2 completes Table D.2, Open Parameters (*), of Minimal OSI.

Table 4-2 RDA Open Parameters

	Referenced Table	Feature	Specification's Statement Sender	Receiver
1	A.6.1 [AARQ]	Calling AE title	**m**	**m**
2		Called AE title	**m**	**m**
3		Calling invocation identifiers	**m**	**m**
4		Called invocation identifiers	**m**	**m**
5		User information	**i**	**i**
6	A.6.2 [AARE]	Responding AE title	**o**	**m**
7		Responding invocation identifiers	**o**	**m**
8		User information	**i**	**i**
9	A.6.3 [RLRQ]	Reason	**i**	**i**
10		User information	**i**	**i**
11	A.6.4 [RLRE]	Reason	**i**	**i**
12		User information	**i**	**i**
13	A.6.5 [ABRT]	User information	**i**	**i**
14	A.7.1 [AARQ and AARE]	Form 1 (Directory name)	**o**	**m**
15		Form 2 (object id + integer)	**m**	**m**
16	B.4.1 [CP]	Default context name	**i**	**i**
17	B.4.3 [CPR]	Default context name	**i**	**i**

4.2 Additional Requirements Imposed by X/Open RDA

This section contains the additional requirements beyond ISO RDA that are imposed by X/Open RDA on use of the OSI upper layers. Since these requirements merely further subset ISO RDA's use of the OSI upper layers, X/Open RDA remains compliant with mOSI.

This section also provides the additional information described in Clause D.2 of Minimal OSI that is needed to complete a profile requirements list for a mOSI-compliant specification.

4.2.1 Additional Parameter Limitations

Table 4-3 specifies additional limits on the values of parameters of the OSI upper-layer services that X/Open RDA imposes.

Table 4-3 Additional Parameter Limitations

	Parameter	Specification's Comment
1	Session Version Number	= version 2
2	Calling Session Selector	4 octets
3	Called Session Selector	4 octets
4	Presentation Protocol Version	= version 1
5	Calling Presentation Selector	4 octets
6	Called Presentation Selector	4 octets
7	ACSE Protocol Version	= version 1

4.2.2 Additional Rules for the Presentation Service

X/Open RDA always selects a value of Fully-encoded-data in User-data, in conjunction with type Single-ASN1-type (see Clause 8.4.2 of Presentation Protocol (see ISO 8823)).

Table 4-4 lists the presentation contexts (abstract syntaxes and associated transfer syntaxes) that X/Open RDA uses in the Presentation Context Definition List parameter.

Table 4-4 Presentation Contexts

Context	Abstract Syntax Object Identifier	Transfer Syntax Object Identifier
ACSE RDA	{2 2 1 0 1} See Section 3.7.1 on page 32.	{2 1 1} {2 1 1}

X/Open RDA does not use a Default Context.

A server informs its clients, in an implementation-defined way, of the entire Presentation Address clients use to establish a connection with that server.

An X/Open-compliant implementation (client or server) can process RDA APDUs of up to 30,000 octets. However, X/Open recommends that implementations do not restrict the size of Presentation user data.

Chapter 5

Non-OSI Transport Providers

This chapter specifies how X/Open RDA permits the use of non-OSI transport providers; that is, the use of message transport facilities in addition to those specified in the ISO Open Systems Interconnection (OSI) series of standards.

5.1 Concepts

This section describes how the OSI model is extended by X/Open RDA to permit use of a variety of transport providers.

5.1.1 Open Systems Interconnection Layered Model

The full OSI Model (see ISO 7498) is a layered model of communication, having the seven layers shown in Figure 5-1.

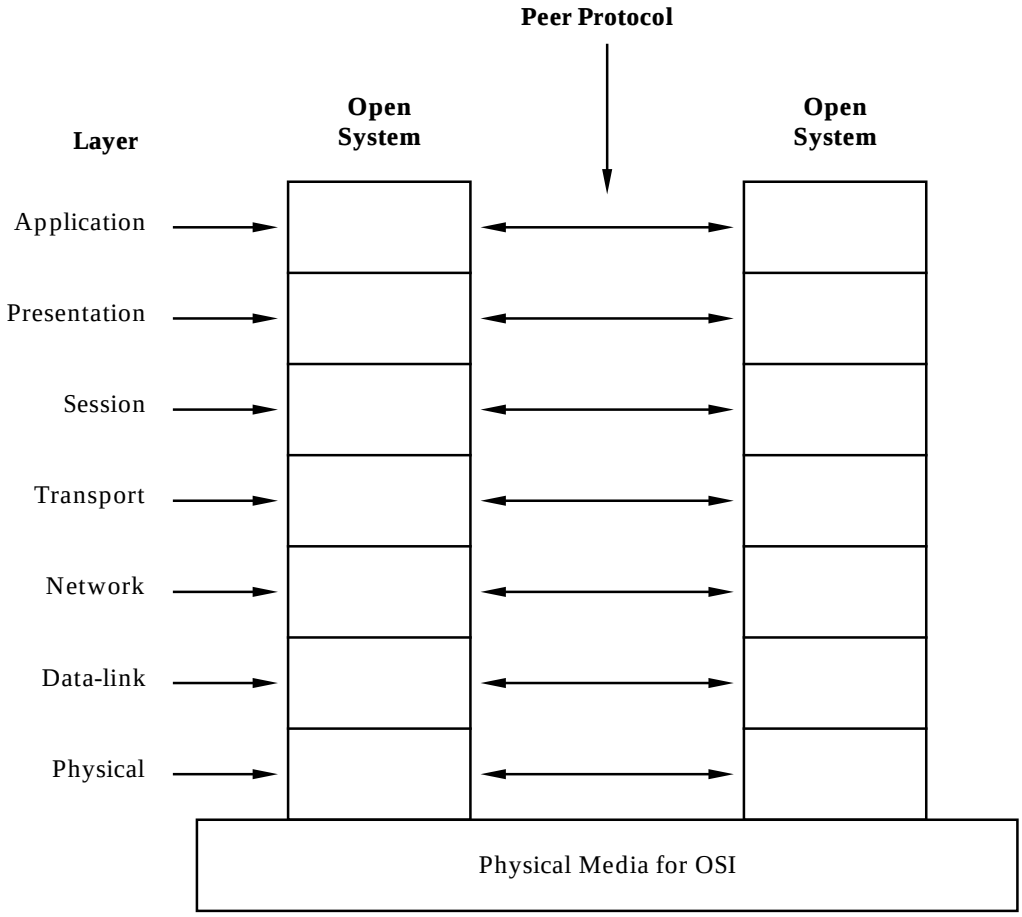

Figure 5-1 OSI Layered Communication Model

With respect to this model, RDA is in the Application Layer.

As expressed in the RFC 1006, a layer is defined by three specifications:

- services offered: the services provided by that layer and the interfaces it offers to access those services
- services required: the services used by that layer and the interfaces it uses to access those services
- protocol: the rules that the layer uses to communicate with the corresponding layer in another system, including the formats for the messages (protocol-data-units, PDUs) that it exchanges with the other system. (This is called the layer's *peer protocol*.)

Thus service definitions define how a single system operates; protocol specifications define how one system operates with another.

5.1.2 Interoperability and Layer Independence

The primary objective of the OSI architecture is interoperability; the ability of one system to operate with another. Interoperability requires agreement on the protocols to be used by the communicating systems; that is, on the messages to be exchanged and the rules for using them (including agreement on which optional elements may be used).

Another motivation for the OSI layered architecture is layer independence: if the services offered by a layer are preserved, then the user of that layer's service is independent of the protocol used by that layer and of the service definitions and protocols used by all lower layers.

With respect to the OSI seven layer model, OSI requires that there is agreement by each layer on its peer protocol. Thus OSI standards specify protocol concretely, with message formats defined (eventually) down to the bit.

However, OSI does not require that a system explicitly separate each of the OSI layers, as long as the system obeys the message formats, and rules for their use, required by those layers. That is, no matter how the OSI layers are implemented (or combined), each layer (or combination of layers) must preserve the message contents of the layers above it.

5.1.3 Non-OSI Transport Providers

There exist a number of non-OSI inter-system communication products that provide message delivery over a variety of physical media. One such is TCP/IP, the Internet's Transmission Control Protocol/Internet Protocol (RFC 793 (see TCP) and RFC 791 (see IP)). The functions of these message delivery facilities are generally comparable to those specified for the Transport Layer in the Transport Service (see ISO 8072).

As a consequence of the desire to provide OSI applications over these types of transport facilities, there has arisen at least one specification for mapping OSI Transport onto such a transport facility, namely the Internet RFC 1006, the ISO Transport Service on top of the TCP (see RFC 1006). By the principle of layer independence, an OSI application that employs the OSI Transport Service will be independent of whether the transport provider is OSI conformant or is some non-OSI transport provider (such as RFC 1006 on TCP).

Such a non-OSI transport facility may not have layers corresponding to the OSI lower layers (Transport, Network, Data-link and Physical), and of course does not provide peer protocols for those lower layers that are compatible with those of OSI. However, this will be invisible to OSI applications and to the OSI upper layers as long as the transport facility preserves the protocols (specifically, the message formats) of the OSI layers above the Transport Layer.

5.1.4 Model for Non-OSI Transport Providers

The model for X/Open RDA, showing the capability for non-OSI transport providers, is given in Figure 5-2 on page 42.

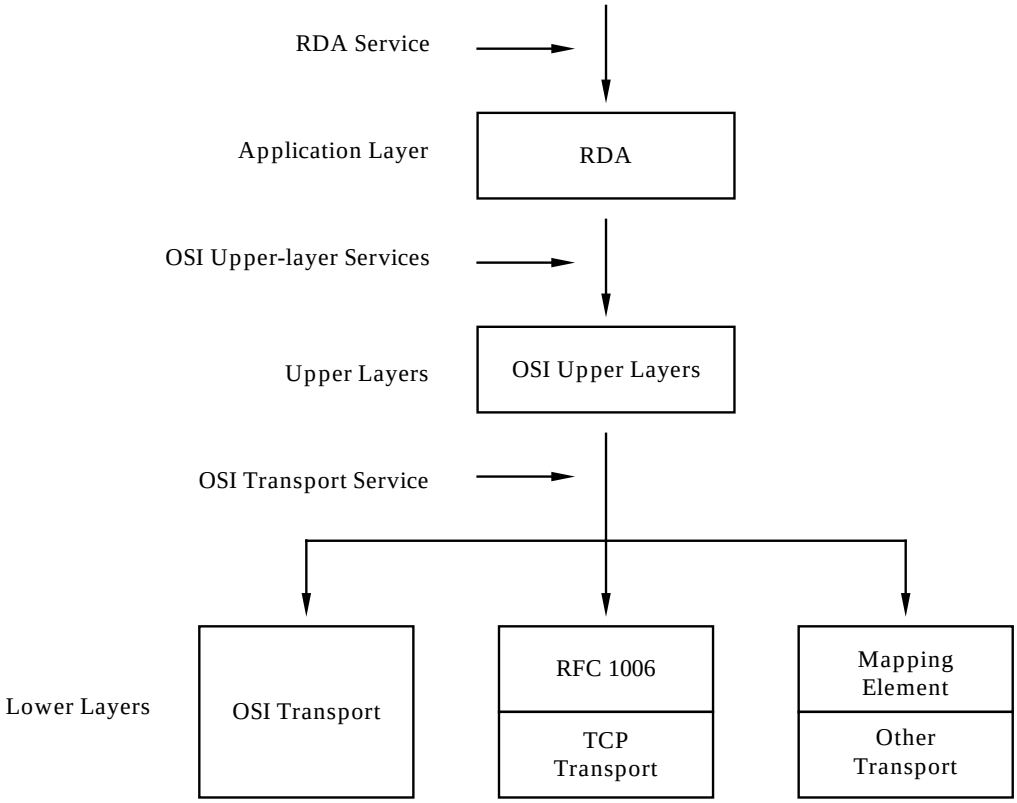

Figure 5-2 Model of RDA for OSI and Non-OSI Transport Providers

In this architecture, the boundary between the OSI upper layers and the lower layers is specified as the OSI Transport Service (see ISO 8072). Above this boundary, the standard OSI Application Layer and upper layers are required. Below this boundary, there is a different set of lower layers for each different type of transport provider. Each such transport provider must specify the mapping from the OSI Transport Service to its particular transport service. It must also specify its protocol, including how Session-protocol-data-units (Transport-service-data-units) are transmitted within its protocol-data-units. (In the case of OSI Transport, no mapping element is required.)

(A visible boundary between the Application Layer and the OSI upper layers is not required to permit use of non-OSI transport providers. It is shown here because RDA is specified, in part, in terms of the services it requires of the OSI upper layers.)

By making the OSI Transport Layer boundary visible and standardisable, this architecture achieves the following objective:

It permits a single implementation of RDA to operate with a variety of transport types.

(However, this architecture does not address how to provide interoperability among transport providers that utilise differing transport protocols.)

5.2 Requirements

5.2.1 Functional Requirements

This section specifies those requirements of X/Open RDA that permit use of non-OSI transport providers. These requirements apply to all transport types; the specific rules for specific transport types are stated in appendices.

- The protocols specified by the OSI standards for the upper layers — RDA, ACSE, Presentation and Session — shall be complied with.

  This means that the OSI Session-PDUs used by RDA shall be presented to the transport provider (as Transport user-data) as specified by the Session Protocol (see ISO 8327). It also means that the PDUs presented to the transport provider shall be the same for all transport types.

- The transport provider shall provide or emulate the services of the OSI Transport Service.

The applicable OSI protocols and services are specified as follows:

RDA — This document, which incorporates by reference and augments the ISO RDA standards RDA Generic and RDA SQL Specialization. Chapter 2 specifies the changes made to the ISO RDA standards for X/Open RDA. Chapter 3 specifies the implementation agreements for X/Open RDA, in particular the permitted values of fields in RDA APDUs. Chapter 4 specifies the requirements imposed by X/Open RDA on the OSI upper layers.

ACSE — The ACSE Procotol (see ISO 8650).

Rules governing the interaction of ISO RDA and ACSE are stated in Clause 5.1.4.1, Association Establishment and Release, Clause 5.1.6, Use of Optional Features, and Clause 5.1.4.2.1, R-Initialize, of RDA Generic.

Additional rules for the use of ACSE by X/Open RDA are given in Section 4.2 on page 38.

Presentation — The Presentation Protocol (see ISO 8823).

Additional rules for the use of the Presentation Service by ISO RDA are given in Clause 5.1.4.3.2, RDA APDUs, and Clause 4.2.3, Concatenation, of RDA Generic.

Additional rules for the use of the Presentation Service by X/Open RDA are given in Section 4.2 on page 38.

Session — The Session Protocol (see ISO 8327).

Additional rules for the use of the Session Service by X/Open RDA are given in Section 4.2 on page 38.

Transport — The Transport Service (see ISO 8072).

5.2.2 Specification Requirements

This section contains the requirements of X/Open RDA upon the specification of the transport provider.

- For each transport type, there shall be one and only one specification of the mapping element (that is, of the element that maps from the OSI Transport Service to that transport type). That specification shall be defined by the authority for that transport type.
- The specification of the mapping element shall include:
  - — the mapping from the OSI Transport Service to the services provided by that transport type
  - — the mapping element's protocol, including how Session-protocol-data-units (Transport-service-data-units) are transmitted within the mapping element's protocol-data-units.

Appendix A

Resulting Text of Table 10

Parts of Chapter 2 of this document specify changes to Table 10 (R-ExecuteDBL use of SQL argument and result parameters) of RDA SQL Specialization. This appendix shows the resulting text of Table 10. Change bars in the right margin identify the X/Open changes.

Table 10. R-Execute use of SQL argument and result parameters.

RDA SQL Statement to be Executed	**ArgSpec**	**ArgVal**	**ResSpec**	**ResVal**
<allocate descriptor statement>[1]				
<alter table statement>				
<close statement>				
<commit statement>[1]				
<connect statement>[1]				
<create index statement>				
<deallocate descriptor statement>[1]				
<declare cursor>	C → S (H) [6]			
<delete statement: positioned>				
<delete statement: searched>	C → S (H)	C → S (H)		
<describe statement>	[11]		C ← S [7]	
<disconnect statement>[1]				
<drop index statement>				
<drop table statement>				
<drop view statement>				
<dynamic close statement>				
<dynamic declare cursor>				
<dynamic delete statement: positioned>				
<dynamic fetch statement>			C → S [8,10]	C ← S
<dynamic open statement>	C → S (H) [10]	C → S (H)	C ← S [9]	
<dynamic update statement: positioned>	C → S (H)	C → S (H)		
<execute statement>	C → S (H) [10]	C → S (H)		
<execute immediate statement>	C → S	C → S		
<fetch statement>			C → S [2]	C ← S
<get descriptor statement>[1]				
<get diagnostics statement>[1]				
<grant statement>				
<insert statement>	C → S (H)	C → S (H)		
<open statement>	C → S (H) [5]	C → S (H)	C ← S [3]	
<prepare statement>	C → S	C → S		
<rollback statement>[1]				
<revoke statement>				
<select statement: single row>	C → S (H)	C → S (H)	C ←→ S [4]	C ← S
<set connection statement>[1]				
<set descriptor statement>[1]				
<table definition>				
<update statement: positioned>	C → S (H)	C → S (H)		
<update statement: searched>	C → S (H)	C → S (H)		
<view definition>				

Legend:	
ArgSpec	The `sQLDBLArgumentSpecification` parameter.
ArgVal	The `sQLDBLArgumentValues` parameter.
ResSpec	The `sQLDBLResultSpecification` parameter.
ResVal	The `sQLDBLResultValues` parameter.
C → S	Parameter supplied by the client.
C ← S	Parameter returned by the server.
(H)	The client must set this parameter if the RDA SQL statement contains host variables, except as noted below.
blank cells	unspecified (that is, not subject to conformance testing)

Notes:	
1	See the `sQLDBLStatementNotAllowed` error under the Error Rules below.
2	The client may optionally send the ResSpec, which will supersede the ResSpec sent on a previously executed (via an R-ExecuteDBL operation) <fetch statement> using the same cursor or received from the server on the corresponding <open statement>.
3	The ResSpec describes the ResVal that the subsequently executed (via an R-ExecuteDBL operation) <fetch statement> using this cursor will return, but any ResSpec specified with such a <fetch statement> overrides this ResSpec.
4	The client may optionally send the ResSpec, otherwise the server shall return the ResSpec which describes the ResVal, if any.
5	The client may optionally send the ArgSpec. If it is not specified, the server shall use the ArgSpec sent on the previously executed (via an R-ExecuteDBL operation) <declare cursor> statement that used the same cursor name.
6	The client may optionally send the ArgSpec. If it is not specified, the client shall send the ArgSpec on the subsequently executed (via an R-ExecuteDBL operation) <open statement> that uses the same cursor name.
7	The RDA server shall return a ResSpec only if the <SQL statement name> in the <describe statement> refers to a prepared statement that is a <cursor specification>. The *n*th SQLDataTypeDescriptor of the ResSpec contains the description of the *n*th column of the table defined by that prepared statement.
8	The client may optionally send the ResSpec, which will supersede the ResSpec sent on a previously executed (via an R-ExecuteDBL operation) <dynamic fetch statement> using the same cursor or received from the server on the corresponding <dynamic open statement>.
9	The ResSpec describes the ResVal that the subsequently executed (via an R-ExecuteDBL operation) <dynamic fetch statement> using this cursor will return, but any ResSpec specified with such a <dynamic fetch statement> overrides this ResSpec. The *n*th SQLDataTypeDescriptor of the ResSpec contains the description of the *n*th column of the table defined by the prepared statement associated with the <dynamic cursor name> referenced in the <dynamic open statement>.
10	If this statement contains a <using clause> that is a <using arguments>, then the RDA client sends an ArgSpec or ResSpec that contains one SQLDataTypeDescriptor for each <embedded variable name> in the <using clause>. The *n*th SQLDataTypeDescriptor corresponds to the *n*th <embedded variable name>. If this statement contains a <using clause> that is a <using descriptor>, then the RDA client sends an ArgSpec or ResSpec that contains one SQLDataTypeDescriptor for each item descriptor area in the SQL descriptor area referenced by the <using descriptor>. If this statement does not contain a <using clause>, then the RDA client does not send any ArgSpec or ResSpec.
11	No ArgSpec is sent by the RDA client, even if the <describe statement> contains a <using descriptor> clause whose <descriptor name> is a reference to an <embedded host variable>.

Appendix B

ASN.1 Module with X/Open Changes

Parts of Chapter 2 of this document specify changes to the ASN.1 module of RDA SQL Specialization. This appendix contains the resulting ASN.1 module. Change bars in the right margin identify the X/Open changes.

```
XOPEN-RDASQL { iso (1) national-member-body (2) uk (826) national (0)          |
                 x-open (1050) data-management (3) x-open-rda (2)              |
                 module (0) version-1 (1) }                                    |
--
-- ********************************************************************
--
DEFINITIONS IMPLICIT TAGS ::= BEGIN
--
-- ********************************************************************
--
IMPORTS
--
  AP-title,
  AE-qualifier,
  AP-invocation-identifier,
  AE-invocation-identifier
  FROM ACSE-1 { joint-iso-ccitt  standard acse (8650) }
--
;

-- ********************************************************************
--
-- RDA SQL APDUs
--

-- top-level 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 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,
    sQLInitializeArgument           [30] SQLInitializeArgument  OPTIONAL
    }

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

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

-- ********************************************************************

R-Terminate-Request              ::= NULL

R-Terminate-Result               ::= NULL

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

-- ********************************************************************

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
    }

R-Cancel-Result                 ::= NULL

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

```
-- ********************************************************************

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

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

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

-- ********************************************************************

R-Open-Request                  ::= SEQUENCE
  { dataResourceHandle              [0] DataResourceHandle,
    dataResourceName                [2] VisibleString OPTIONAL,
    sQLAccessControlData            [3] AccessControlData OPTIONAL,
    sQLUsageMode                    [4] SQLUsageMode DEFAULT retrieval,
    sQLOpenArgument                [30] SQLOpenArgument  OPTIONAL
    }

R-Open-Result                   ::= SEQUENCE
  { sQLOpenResult                  [30] SQLOpenResult OPTIONAL
    }

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

```
-- ********************************************************************

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

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

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

-- ********************************************************************

R-ExecuteDBL-Request            ::= SEQUENCE
  { dataResourceHandle              [0] DataResourceHandle  OPTIONAL,
    sQLDBLStatement                 [1] SQLDBLStatement,
    sQLDBLArgumentSpecification     [2] SQLDBLArgumentSpecification  OPTIONAL,
    sQLDBLResultSpecification       [3] SQLDBLResultSpecification  OPTIONAL,
    dblArguments                    CHOICE
    { singleArgument                [4] SEQUENCE
      { repetitionCount                 [0] INTEGER DEFAULT 1,
        sQLDBLArgumentValues            [1] SQLDBLArgumentValues  OPTIONAL
        },
      multipleArgument              [5] SEQUENCE
      { listOfSQLDBLArgumentValues      [0] SEQUENCE OF SQLDBLArgumentValues
        }
      }                                 OPTIONAL,
    returnSQLDiagnostics            [6] BOOLEAN DEFAULT FALSE                |
    }

R-ExecuteDBL-Result             ::= SEQUENCE
  { sQLDBLResultSpecification       [1] SQLDBLResultSpecification  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,
    sQLExecuteDBLError              SQLExecuteDBLError
    }
```

```
-- ********************************************************************

R-DefineDBL-Request              ::= SEQUENCE
  { commandHandle                    [0] CommandHandle,
    dataResourceHandle               [1] DataResourceHandle  OPTIONAL,
    sQLDBLStatement                  [2] SQLDBLStatement,
    sQLDBLArgumentSpecification      [3] SQLDBLArgumentSpecification  OPTIONAL,
    sQLDBLResultSpecification        [4] SQLDBLResultSpecification  OPTIONAL
    }

R-DefineDBL-Result               ::= SEQUENCE
  { sQLDBLResultSpecification        [0] SQLDBLResultSpecification  OPTIONAL,
    sQLDBLException                  [1] SQLDBLException  OPTIONAL
    }

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

-- ********************************************************************

R-InvokeDBL-Request              ::= SEQUENCE
  { commandHandle                    [0] CommandHandle,
    dblArguments                     CHOICE
    { singleArgument                 [1] SEQUENCE
      { repetitionCount                  [0] INTEGER DEFAULT 1,
        sQLDBLArgumentValues             [1] SQLDBLArgumentValues  OPTIONAL
        },
      multipleArgument               [2] SEQUENCE
      { listOfSQLDBLArgumentValues       [0] SEQUENCE OF SQLDBLArgumentValues
        }
      }                                OPTIONAL
    }

R-InvokeDBL-Result               ::= SEQUENCE
  { sQLDBLResultSpecification        [0] SQLDBLResultSpecification  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,
    sQLInvokeDBLError                SQLInvokeDBLError
    }
```

```
-- ********************************************************************

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

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

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

-- ********************************************************************
--
-- 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
      }
    }

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
      }
    }
```

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

OperationID                     ::= INTEGER

ResultValues                    ::= SEQUENCE
  { sQLDBLException                 [0] SQLDBLException,
    sQLDBLResultValues              [1] SQLDBLResultValues  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
      }
    }

-- ********************************************************************
```

```
--
-- Definitions of generic ASN.1 errors
--

AccessControlViolation          ::= [APPLICATION  0] NULL

BadRepetitionCount              ::= [APPLICATION  1] NULL

CommandHandleUnknown            ::= [APPLICATION  2] NULL

ControlAuthenticationFailure    ::= [APPLICATION  3] NULL

ControlServicesNotAllowed       ::= [APPLICATION  4] NULL

DataResourceAlreadyOpen         ::= [APPLICATION  5] SEQUENCE
  { dataResourceHandle              [0] DataResourceHandle
    }

DataResourceHandleNotSpecified  ::= [APPLICATION  6] NULL

DataResourceHandleUnknown       ::= [APPLICATION  7] NULL

DataResourceNameNotSpecified    ::= [APPLICATION  8] NULL

DataResourceNotAvailable        ::= [APPLICATION  9] ErrorDiagnostic

DataResourceUnknown             ::= [APPLICATION 10] NULL

DialogueIDUnknown               ::= [APPLICATION 11] NULL

DuplicateCommandHandle          ::= [APPLICATION 12] NULL

DuplicateDataResourceHandle     ::= [APPLICATION 13] NULL

DuplicateDialogueID             ::= [APPLICATION 14] NULL

DuplicateOperationID            ::= [APPLICATION 15] NULL

InvalidSequence                 ::= [APPLICATION 16] SEQUENCE
  { diagnosticInformation           [0] ENUMERATED
    { dialogueNotActive             (1),
      dialogueInitializing          (2),
      dialogueAlreadyActive         (3),
      transactionNotOpen            (4),
      transactionOpen               (5),
      transactionTerminating        (6),
      dialogueTerminating           (7)
      }                             OPTIONAL
    }

NoDataResourceAvailable         ::= [APPLICATION 17] NULL

OperationAborted                ::= [APPLICATION 18] ErrorDiagnostic

OperationCancelled              ::= [APPLICATION 19] NULL

ServiceNotNegotiated            ::= [APPLICATION 20] NULL

TransactionRolledBack           ::= [APPLICATION 21] NULL
```

```
UserAuthenticationFailure       ::= [APPLICATION 22] NULL

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

-- ********************************************************************
```

```
--
-- Definitions of SQL Specialization parameters
--

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

ExceptionInfo                   ::= SEQUENCE                                |
  { returnedSQLSTATE                [0] VisibleString,                        |
    classOrigin                     [1] VisibleString,                        |
    subclassOrigin                  [2] VisibleString,                        |
    messageText                     [3] VisibleString OPTIONAL                |
    }                                                                         |

SQLDataTypeDescriptor           ::= SEQUENCE
  { nullable                        [0] BOOLEAN DEFAULT TRUE,
    colName                         [1] VisibleString OPTIONAL,
    typeDescriptor                  CHOICE
    { characterType                 [5] SEQUENCE
      -- SQL Type: character
      { charSet                         OBJECT IDENTIFIER OPTIONAL,
        length                          INTEGER,
        fixedLengthEncoding             BOOLEAN
        },
      numericType                   [6] SEQUENCE
      -- SQL Type: numeric
      { precision                       INTEGER,
        scale                           INTEGER
        },
      decimalType                   [7] SEQUENCE
      -- SQL Type: decimal
      { precision                       INTEGER,
        scale                           INTEGER
        },
      integerType                   [8] SEQUENCE
      -- SQL Type: integer
      { precision                       INTEGER,
        precisionBase                   ENUMERATED
        { binary                        (0),
          decimal                       (1)
          }
        },
      smallIntType                  [9] SEQUENCE
      -- SQL Type: smallInt
      { precision                       INTEGER,
        precisionBase                   ENUMERATED
        { binary                        (0),
          decimal                       (1)
          }
        },
      floatType                    [10] SEQUENCE
      -- SQL Type: float
      { mantissaPrecision               INTEGER,
        maxExponent                     INTEGER
        },
      realType                     [11] SEQUENCE
```

```
      -- SQL Type: real
      { mantissaPrecision               INTEGER,
        maxExponent                     INTEGER
        },
      doublePrecisionType           [12] SEQUENCE
      -- SQL Type: doublePrecision
      { mantissaPrecision               INTEGER,
        maxExponent                     INTEGER
        },
      varcharType                   [15] SEQUENCE                                |
      -- SQL Type: varchar                                                        |
      { charSet                          OBJECT IDENTIFIER OPTIONAL,             |
        length                           INTEGER                                 |
        }                                                                        |
      }
    }

SQLDBLArgumentSpecification     ::= SEQUENCE
  { listOfSQLDataTypeDescriptor     [0] SEQUENCE OF SQLDataTypeDescriptor
    }

SQLDBLArgumentValues            ::= SQLValueList

SQLDBLException                 ::= SEQUENCE
  { sQLSTATE                        [0] VisibleString OPTIONAL,
    sQLCODE                         [1] INTEGER OPTIONAL,
    sQLErrorText                    [2] VisibleString OPTIONAL,
    sQLDiagnostics                  [3] SQLDiagnostics OPTIONAL                  |
    }

SQLDBLResultSpecification       ::= SEQUENCE
  { listOfSQLDataTypeDescriptor     [0] SEQUENCE OF SQLDataTypeDescriptor
    }

SQLDBLResultValues              ::= SQLValueList

SQLDBLStatement                 ::= SEQUENCE
  { statementText                   [0] OCTET STRING,
    charSet                         [1] OBJECT IDENTIFIER OPTIONAL
    }

SQLDiagnostics                  ::= SEQUENCE                                     |
  { rowCount                        [0] INTEGER OPTIONAL,                        |
    exceptionList                   [3] SEQUENCE OF ExceptionInfo                |
    }                                                                            |

SQLInitializeArgument           ::= SEQUENCE
  { sQLConformanceLevelDefault      [0] OBJECT IDENTIFIER OPTIONAL,
    userData                        [1] OCTET STRING OPTIONAL
    }

SQLInitializeResult             ::= SEQUENCE
  { userData                        [0] OCTET STRING OPTIONAL
    }

SQLOpenArgument                 ::= SEQUENCE
  { charSet                         [0] OBJECT IDENTIFIER OPTIONAL,
    sQLConformanceLevel             [1] OBJECT IDENTIFIER OPTIONAL,
    sQLDiagnosticsRequested         [2] ENUMERATED                               |
```

```
    { always                        (0),                                    |
      onRequest                     (1),                                    |
      never                         (2)                                     |
      } DEFAULT never                                                       |
    }

SQLOpenResult                   ::= SEQUENCE
  { charSet                         [0] OBJECT IDENTIFIER OPTIONAL,
    charSetNotSupported             [1] BOOLEAN DEFAULT FALSE,
    sQLConformanceLevel             [2] OBJECT IDENTIFIER OPTIONAL
    }

SQLUsageMode                    ::= ENUMERATED
  { retrieval                       (0),
    update                          (1)
    }

SQLValue                        ::= SEQUENCE
  { dataItem                        CHOICE
    { characterItem                 [0] OCTET STRING,
      numericItem                   [1] INTEGER,
      decimalItem                   [2] INTEGER,
      integerItem                   [3] INTEGER,
      smallIntItem                  [4] INTEGER,
      floatItem                     [5] REAL,
      realItem                      [6] REAL,
      doublePrecisionItem           [7] REAL,
      varcharItem                  [10] OCTET STRING                      |
      }                             OPTIONAL,
    indicator                      [30] INTEGER OPTIONAL
    }

SQLValueList                    ::= CHOICE
  { listOfSQLValue                  [1] SEQUENCE OF SQLValue
    }

-- ********************************************************************
```

```
--
-- Definitions of RDA SQL Specialization Errors
--

HostIdentifierError             ::= [APPLICATION 23] NULL

InvalidSQLConformanceLevel      ::= [APPLICATION 24] NULL

RDATransactionNotOpen           ::= [APPLICATION 25] NULL

RDATransactionOpen              ::= [APPLICATION 26] NULL

SQLAccessControlViolation       ::= [APPLICATION 27] NULL

SQLDatabaseResourceAlreadyOpen  ::= [APPLICATION 28] NULL

SQLDBLArgumentCountMismatch     ::= [APPLICATION 29] NULL

SQLDBLArgumentTypeMismatch      ::= [APPLICATION 30] NULL

SQLDBLNoCharSet                 ::= [APPLICATION 31] NULL

SQLDBLStatementNotAllowed       ::= [APPLICATION 32] NULL                    |

SQLUsageModeViolation           ::= [APPLICATION 33] NULL

--
-- SQL Specialization errors which can be returned for specific RDA operations
--

SQLCloseError                   ::= CHOICE
  { rDATransactionOpen              RDATransactionOpen
    }

SQLDefineDBLError               ::= CHOICE
  { hostIdentifierError             HostIdentifierError,
    sQLDBLNoCharSet                 SQLDBLNoCharSet,
    sQLDBLStatementNotAllowed       SQLDBLStatementNotAllowed,               |
    sQLUsageModeViolation           SQLUsageModeViolation
    }

SQLExecuteDBLError              ::= CHOICE
  { hostIdentifierError             HostIdentifierError,
    rDATransactionNotOpen           RDATransactionNotOpen,
    sQLDBLArgumentCountMismatch     SQLDBLArgumentCountMismatch,
    sQLDBLArgumentTypeMismatch      SQLDBLArgumentTypeMismatch,
    sQLDBLNoCharSet                 SQLDBLNoCharSet,
    sQLDBLStatementNotAllowed       SQLDBLStatementNotAllowed,               |
    sQLUsageModeViolation           SQLUsageModeViolation
    }
```

```
SQLInvokeDBLError                ::= CHOICE
  { rDATransactionNotOpen            RDATransactionNotOpen,
    sQLDBLArgumentCountMismatch      SQLDBLArgumentCountMismatch,
    sQLDBLArgumentTypeMismatch       SQLDBLArgumentTypeMismatch,
    sQLUsageModeViolation            SQLUsageModeViolation
    }

SQLOpenError                     ::= CHOICE
  { invalidSQLConformanceLevel       InvalidSQLConformanceLevel,
    rDATransactionOpen               RDATransactionOpen,
    sQLAccessControlViolation        SQLAccessControlViolation,
    sQLDatabaseResourceAlreadyOpen   SQLDatabaseResourceAlreadyOpen
    }

-- ********************************************************************
END           -- RDA SQL ASN.1 Module
-- ********************************************************************
```

Appendix C

X/Open RDA for OSI Transport

This appendix specifies how X/Open RDA is provided on OSI transport providers. Refer to Chapter 5 on page 39 for the concepts and general requirements for all transport providers.

- The defining authority for OSI Transport is the International Standards Organization (ISO).
- No mapping element is required.
- The transport protocol is the OSI Transport Protocol, as specified in the Transport Protocol (see ISO/IEC 8073).

Appendix D

X/Open RDA for TCP Transport

This appendix specifies how X/Open RDA is provided on TCP transport providers. Refer to Chapter 5 on page 39 for the concepts and general requirements for all transport providers.

- The defining authority for TCP transport is the Internet Engineering Task Force (IETF).
- The mapping element is the ISO Transport Service on top of the TCP, as specified in RFC 1006.
- The transport protocol is the Transmission Control Protocol (TCP), as specified in RFC 793 (see TCP).

Appendix E

Application Programming Interfaces

This appendix is informative.

This appendix describes how the X/Open RDA model permits:

- use of a subset of the full OSI upper-layer services
- employment of standardised interfaces both to those upper-layer services and to the Transport Layer services.

It also contains X/Open's recommendations for such interfaces and guidance on their usage by X/Open RDA.

E.1 Concepts

This section describes how the X/Open RDA model (as shown in Figure 5-2 on page 42) is further extended from the OSI model in order to identify standardisable interfaces.

E.1.1 Portability

Portability is the ability to implement a set of services (a product) easily on a variety of systems. Portability requires agreement on the interfaces to be used between layers; that is, on the specifications of how to invoke the services offered by a layer.

Interoperability, discussed in Section 5.1.2 on page 41, and portability are different objectives, with different relationships to the OSI model. Portability is not required by OSI; how a system provides its services is defined abstractly but not concretely (in OSI, concrete specification of a service is a *local matter*). In other words, OSI standards do not specify application programming interfaces (APIs).

E.1.2 Minimal OSI Facility

The following factors have motivated the development of the *minimal OSI upper layers* (mOSI) facility (see Minimal OSI):

- Many OSI applications require only a subset of the full services of the OSI upper layers (the *services required* by an OSI application of commonly used Application Layer services, the Presentation Service, and the Session Service).
- The OSI upper layers are complex, with potentially high cost in space and speed.
- The principle of layer independence, together with the fact that OSI service definitions are abstract, permits OSI layers to be combined.
- As noted in Section 5.1.3 on page 41, there exist several types of message delivery facilities with services comparable to OSI Transport. Thus maintaining the boundary to the Transport Layer, and providing a visible, standardised interface specification for the Transport Service, has value.

These factors have led to the definition of mOSI as a subset of the services of the OSI upper layers: the Association Control Service Element (although this is an Application Layer service element, it is used by many OSI Application Layer services), the Presentation Layer, and the Session Layer. The mOSI subset is defined as an International Standardized Profile of the OSI upper layers.

Since mOSI is a subset of the OSI upper layers, an application using a mOSI implementation of the OSI upper layers can interoperate with an application using a full implementation of the OSI upper layers.

As described in Chapter 4 on page 35, mOSI provides sufficient OSI upper-layer services for X/Open RDA.

E.1.3 Programming Interfaces

Although, as stated earlier, portability is not an OSI objective, it is an X/Open objective, and hence X/Open desires the definition and use of standardised programming interfaces (APIs). There are several interfaces (that is, several levels in the layered OSI model) at which a standardised API might be specified in order to enhance the portability of service-users of that layer:

- interfaces to the Application Layer service — for RDA, this would be an API for an RDA service-provider
- interfaces to the upper layer services (common Application Layer services, Presentation and Session)
- interfaces to the lower-layer services (Transport and below).

E.1.4 Model for Standardised APIs

The resultant model of X/Open RDA is given in Figure E-1. This model augments that shown in Figure 5-2 on page 42 to show the ability to use mOSI and to use standardised APIs.

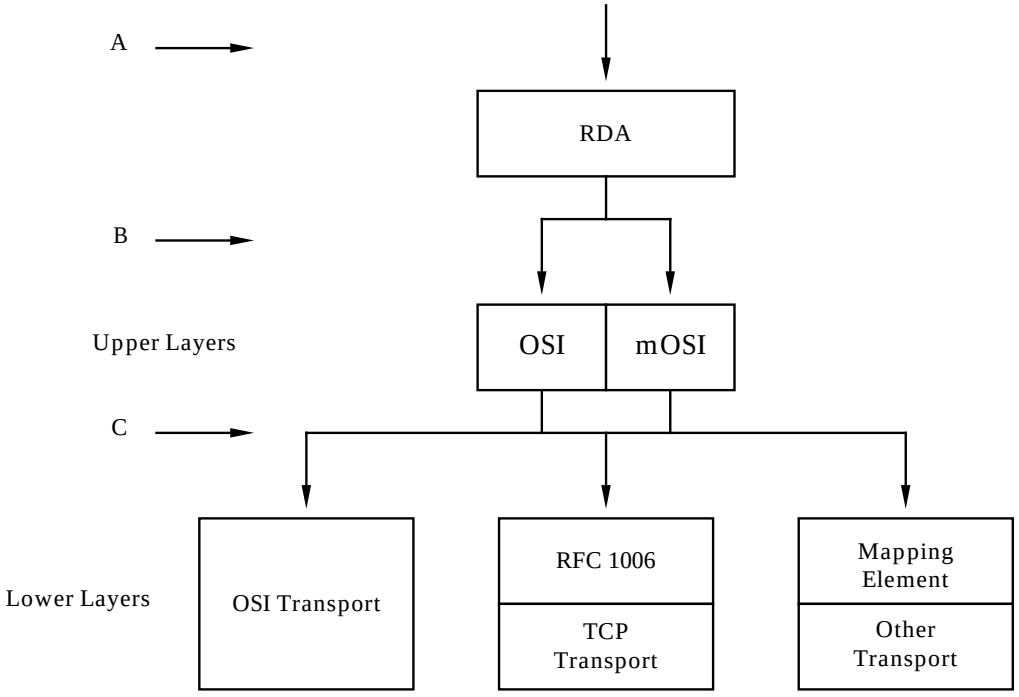

Figure E-1 Model of RDA Using Standardised APIs

By making certain interfaces visible and standardisable, this architecture achieves the following objectives:

- It permits a single implementation of RDA to operate with either of two implementations of the OSI upper layers — full OSI or minimal OSI.
- It permits these two implementations of the OSI upper layers to interoperate.
- It permits both of these implementations of the OSI upper layers to operate with the various transport types.

The three interfaces identified in this model are:

A: The interface to an RDA service-provider. This document does not provide or recommend an API for this interface.

B: The interface to an OSI upper-layer services provider (ACSE, Presentation and Session). This document recommends either of two APIs for this interface:

1. The X/Open ACSE/Presentation Services API (XAP), as specified in X/Open **XAP** specification. This interface is recommended for vendors that wish to implement the full set of OSI upper-layer services.
2. The X/Open XTI interface to mOSI (XTI-mOSI), as specified in the X/Open **mOSI Functionality** specification. This interface is recommended for vendors that wish to implement only the minimum set of OSI upper-layer services that is required by X/Open RDA (this document).

C: The interface to an OSI-compliant transport service provider. This document recommends the X/Open Transport Interface, as specified in the X/Open **XTI, Version 2** specification and particularly its appendices for various transport providers, for this interface.

This model is consistent with those of:

- X/Open XAP, as given in Figure 1-1, OSI Service Interfaces, of the X/Open **XAP** specification
- ISO Minimal OSI, as given in Figure 1, mOSI Model, in Clause 7 of Minimal OSI.

For additional discussion of the OSI layers and service interfaces and of the roles of XAP, XTI-mOSI and XTI, the reader is referred to:

- Sections 1.1 and 1.4 of the X/Open **XAP** specification
- Clause 7 and Annex H.4 of Minimal OSI.

E.2 Interfaces to the Upper Layers

For the interface to an OSI upper-layers service-provider (ACSE, Presentation and Session), X/Open recommends either:

1. the X/Open ACSE/Presentation Services API (see the X/Open **XAP** specification)

   or:

2. the X/Open XTI interface to mOSI (see the X/Open **mOSI Functionality** specification).

E.2.1 General Guidelines

An RDA service provider (an implementation of RDA) uses the services of the OSI upper layers via the Association Control Service Element and the Presentation Service.

(The rules for the use by X/Open RDA of the OSI upper layers are stated in various clauses of RDA Generic and sections of this X/Open **RDA** specification, and are referenced in Section 5.2 on page 43.)

The following procedure outlines how an X/Open OSI-compliant application (for this document, X/Open RDA) employs the API for the upper-layer services (whether XAP or XTI-mOSI) to provide the services of the OSI ACSE and Presentation Service. The procedure presented is intended solely as a general description of how the APIs might be used, and is not a requirement for X/Open RDA implementations.

1. Establishing (Initialising) an API instance — RDA creates an instance of the API, initialises environment variables, and binds the API instance to a presentation-address.
2. Establishing the association — RDA uses the ACSE A-ASSOCIATE service to establish an association.
3. Sending RDA service requests and responses — RDA uses the Presentation P-DATA service to transmit RDA APDUs.
4. Releasing the association:
   - — Normally — RDA uses the ACSE A-RELEASE service to release an association in an orderly manner.
   - — Abnormally — RDA uses the ACSE A-ABORT and A-P-ABORT services to release an association abnormally.
5. Releasing (De-initialising) an API instance — RDA closes the instance of the API.

The API functions used in steps 2. to 4. to provide the ACSE and Presentation services differ somewhat between XAP and XTI-mOSI as described later in this chapter.

The parameters of ACSE and Presentation that are used by RDA can be derived from the *supported fields* tables in Annex C, ACSE Protocol PDUs, and Annex B, Presentation Protocol PDUs, respectively, of RDA ISP, Part 3. Additional limitations on those parameters that are imposed by X/Open RDA are stated in Section 4.2 on page 38.

E.2.2 Maximising Portability between XAP and XTI-mOSI

There are differences between XAP and XTI-mOSI, both in the OSI upper-layer services and parameters provided and in the way those services are mapped to XAP or XTI-mOSI functions.

To maximise portability between the XAP and XTI-mOSI APIs, an X/Open RDA implementation should use only those OSI upper-layer services that are provided by XTI-mOSI (which are a subset of the full OSI upper-layer services available via XAP), and for those services should use only those parameters that are supported by XTI-mOSI.

Some of the aspects in which XAP and XTI-mOSI differ are:

- the specific functions used to provide the ACSE and Presentation services
- handling of presentation contexts and presentation context identifiers
- handling of OSI addresses.

E.2.3 XAP

X/Open recommends the X/Open XAP interface (see X/Open **XAP** specification) for vendors that wish to implement the full set of OSI upper-layer services.

For guidance on the use of XAP, the reader is referred particularly to the following sections of the X/Open **XAP** specification:

- Section 1.1, Objectives, for the objectives of XAP.
- Section 1.4, Overview of ACSE/Presentation Services, for the OSI upper-layer services provided by XAP.
- Section 2.1, XAP Model, for the relationship of the XAP model to OSI concepts.
- Section 2.3, Using the XAP Interface, for how XAP functions are used to establish an association and transfer application information; this section describes how the procedural steps described in Section E.2.1 on page 77 can be implemented via XAP functions. (In Section 2.3 of the X/Open **XAP** specification the steps are categorised somewhat differently from Section E.2.1. Also, step 5 is not explicitly mentioned; it can be accomplished via the *ap_close*( ) function.)
- Section 2.2, XAP Functions and Mechanisms, for the categorisation and a list of the specific XAP functions.
- Section 2.2.6, Presentation Context Negotiation, for how XAP handles negotiation of presentation contexts. The XAP user (RDA) must specify both presentation contexts required, RDA and ACSE, and must provide presentation context identifiers.
- Section 2.2.7, Presentation Addresses, for how XAP handles addressing.
- Section 2.2.12, User Data Mechanisms, for how XAP handles data encoding and segmentation of encoded data.

With XAP, all of the parameters required by ACSE and Presentation must be supplied by the user of XAP (that is, by RDA).

E.2.4 XTI-mOSI

X/Open recommends the XTI-mOSI interface for vendors that wish to implement only the minimum set of OSI upper-layer services that is required by X/Open RDA.

For guidance on the use of XTI-mOSI, the reader is referred particularly to the following sections of the X/Open **mOSI Functionality** specification:

- Section H.1.1, Rationale for Using XTI-mOSI, for the objectives of XTI-mOSI.
- Section H.1.3, OSI Functionality, for the OSI upper-layer facilities provided by XTI-mOSI.
- Section H.4.3.1, Connection-oriented Services, for how XTI-mOSI functions (and to Section 3.1, Overview of Connection-oriented Mode, of the X/Open **XTI, Version 2** specification for how XTI functions in general) are used to establish an association and transfer application information; these sections describe how the procedural steps described in Section E.2.1 on page 77 can be implemented via XTI-mOSI functions. (In Section H.4.3.1 of the X/Open **mOSI Functionality** specification, only steps 2 through 4 of Section E.2.1 are explicitly mentioned; steps 1 and 5 are described in Section 3.1.1, Initialisation/De-initialisation Phase of the X/Open **XTI, Version 2** specification.)
- Section H.3, Functions, for a list of the specific XTI-mOSI functions.

  RDA requires the Calling AP and AE Invocation-identifiers (of the A-ASSOCIATE service). At present, XTI-mOSI does not support these parameters, although they are included in Minimal OSI. It is expected that these parameters will be added in a future revision of XTI-mOSI.
- Section H.2.1, ACSE/Presentation Connection-oriented Service, for the XTI-mOSI options provided for the OSI upper-layer services. (RDA cannot use connectionless Presentation services.)

  Since RDA uses only its own abstract syntax in addition to that of ACSE, and X/Open RDA uses only one transfer syntax (Basic Encoding Rules), X/Open RDA need not use multiple choice presentation contexts. (XTI-mOSI manages the presentation context of ACSE, as described in Section H.4.2. of the X/Open **mOSI Functionality** specification.)
- Section H.4.2, Mandatory and Optional Parameters, for how the ACSE parameters provided by XTI-mOSI are used.

  The XTI-mOSI user (RDA) must specify only the abstract syntax for RDA, and does not provide presentation context identifiers. (XTI-mOSI provides the presentation context for ACSE, as well as the required presentation context identifiers.)
- Section H.1.5.1, Naming and Addressing Information Used by mOSI, for how XTI-mOSI handles addressing.
- Section H.5, Complements to <xti.h>, for the encoding of OSI upper-layer service parameters, in particular naming and addressing, object identifiers and presentation context definitions.
- Section H.2.3, Transport Service Options, for the XTI options that may be available to users of XTI-mOSI.

  X/Open RDA should not use the quality of service transport options, which are both unavailable with an ISO-over-TCP transport provider and implementation dependent.

With XTI-mOSI, many of the parameters of ACSE and Presentation have default or fixed values, and therefore need not or cannot be supplied by the user of XTI-mOSI (that is, by RDA).

E.3 Interfaces to the Transport Layer

For the interfaces to the (OSI-compliant) transport service provider, X/Open recommends:

> The X/Open Transport Interface, as specified in the X/Open **XTI, Version 2** specification and particularly its appendices for various transport providers.

RDA is not the direct user of the Transport Layer; rather, the upper-layer provider is. Therefore guidance on the use of XTI is more pertinent to implementors of upper-layer services, whether full OSI or mOSI, than to implementors of RDA, and is therefore generally beyond the scope of this document. However, some such information is given below, along with references to the applicable documents, in order to aid the designers of new mapping elements from OSI Transport to additional transport providers.

E.3.1 Minimising Differences Among Transport Interfaces

As noted in Chapter 1, General Introduction to the XTI, of the X/Open **XTI, Version 2** specification, ‘‘While XTI gives transport users considerable independence from the underlying transport provider, the differences between providers are not entirely hidden’’. Appendix C.4 of the X/Open **XTI, Version 2** specification contains guidelines for writing software that is independent of the transport provider, which can be done primarily by:

- using only functions provided by all providers
- avoiding option management
- using a provider-independent means of acquiring addresses.

Some factors to note are:

- the specific values permitted for the maximum length of Transport-PDUs, as specified in TCO_LTPDU
- RDA requires connection-mode transport.

E.3.2 OSI Transport

For the API to an OSI transport provider, X/Open recommends the XTI interface described in Appendix A, ISO Transport Protocol Information, of the X/Open **XTI, Version 2** specification.

Note: The Quality of Service parameters should not be used if the implementor desires to maximise portability over various transport types. They are protocol-specific, and for example not available with TCP.

E.3.3 TCP Transport

For the API to a TCP transport provider, X/Open recommends the XTI interface described in Appendix A, ISO Transport Protocol Information, of the X/Open **XTI, Version 2** specification. That specification assumes the use of RFC 1006, which is the protocol that X/Open RDA requires over TCP transport (see Appendix D on page 71). The resultant service is essentially equivalent to OSI Transport Class 4.

Some additional information pertinent to OSI over TCP transport is contained in Appendix B.1, Internet Protocol-specific Information — General, of the X/Open **XTI, Version 2** specification.

Glossary

ACSE
Association control service element.

APDU
Application protocol data unit.

ASN.1
(Abstract Syntax Notation One) A notation, defined in ASN.1 (see ISO 8824), that allows data to be described in a machine-independent fashion.

Association Control Service Element
The application service element responsible for association establishment and release. All OSI application entities contain an ACSE.

Application Protocol Data Unit
A unit of data specified in an application layer protocol and consisting of application protocol control information and possibly application user data.

byte
An individually addressable unit of data storage that is equal to or larger than an octet.

client
(RDA Client) The RDA service-user that initialises an RDA dialogue and requests database access from a remote database server.

database
(SQL Database Resource) The data and the schemas describing it, as defined in the X/Open **SQL** specification.

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

dynamic SQL
A way to execute SQL statements whose specifics may not be known at compile time.

interoperability
The ability of one system to operate with another.

layer
A subdivision of the OSI architecture, constituted by subsystems of the same level in the OSI layered architecture.

layer independence
The independence of the user of a layer's services from the protocol specification of that layer and from the service definitions and protocol specifications of all lower layers.

local matter
An aspect of operation that an implementation is free to choose without violating the provisions of a standard.

lower layers
The layers in the OSI architecture below the Transport Service boundary; that is, the Transport, Network, Data-link and Physical Layers.

mapping element
A facility that maps the services of the OSI Transport Service to the services offered by some other transport provider. For example, RFC 1006 is a mapping element to TCP transport.

minimal OSI upper layers
A profile of the upper layers that includes only a minimal subset of their functions, as specified in Minimal OSI. Such a profile is sufficient for X/Open RDA.

octet
Eight bits in a row, not necessarily at an addressable machine boundary.

peer protocol
The protocol used between entities within the same layer.

portability
The ability to implement a set of services (a product) easily on a variety of systems.

Presentation Layer
Layer six of the OSI Reference Model; the presentation layer provides the mechanisms for negotiating the common method for representing information and for transferring information so that the semantics are preserved during the transfer.

profile
A definition of a combination of base standards that collectively performs a specific, well-defined function. A profile standardises the use of options and other variations in the standards to which it applies, and thus provides a basis for interoperability.

server
(RDA server) The RDA service-user within a database server that provides database access to remote RDA clients.

services offered
The services provided by a layer to the layer above it and the interfaces it offers to access those services.

services required
The services used by a layer from the layer below it and the interfaces it uses to access those services.

Session Layer
Layer five of the OSI Reference Model; the session layer provides the mechanisms for establishing a session connection between cooperating users and for the organised and synchronised exchange of data between those users.

SQL
(Structured Query Language) A database language widely accepted as an interface to relational database management systems.

Transport Layer
Layer four of the OSI Reference Model; the Transport Layer provides transparent transfer of data between OSI and open systems.

transport provider
A facility that provides services comparable to those of the OSI Transport Layer. Both the

OSI Transport Service and RFC 1006 are examples of transport providers that provide exactly the services of the OSI Transport Layer.

upper layers

The layers in the OSI architecture above the Transport Service boundary; that is, the Application, Presentation and Session Layers.

Index