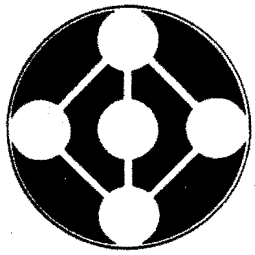

Alternative Technologies

IDM - RS/1 INTERFACE USERS MANUAL

PREPARED BY
DAVID MCGOVERAN

CONTENTS

1. INTRODUCTION

The IDM-RS/1 interface enables you to access, from RS/1, information stored in the IDM database. You interface with the IDM by means of program calls within RS/1.

The IDM-RS/1 interface requires two types of communication areas:

Logon Data Area (LDA)

Communication Data Areas (CDA)

There is one LDA for each RS/1 program which accesses IDM. The LDA is a 64-byte data area (within the RS/1 program) which the interface uses to facilitate communication between IDM and RS/1. You set up the LDA by issuing the O$LON call (described below) within your RS/1 program. Once you have set up the LDA, the IDM-RS/1 interface manages the LDA for you.

The CDA is the name of a 64-byte data area (within the RS/1 program) which the interface uses to identify and control an active query statement. There is one CDA associated with each active query statement. Each CDA controls only one query statement at a time (after you have closed a CDA, the interface can use that CDA to control another query statement). You can have a maximum of 20 active query statements at the same time in a single program (you can change this maximum number by using the O$SETPARAMS function, described below). You establish a CDA by issuing an O$OPEN call from within your RS/1 program. Once you have established a CDA data area, RS/1 manages it for you; you need only use the CDA-id (returned by the O$OPEN call) to tell RS/1 which CDA you are using. A CDA-id is similar to an RS/1 channel-id (CHNID), which is used to reference open files. It is also, in fact, an IDM runtime control block.

Not all data types are supported between the IDM and RS/1; however, those which are supported cover most of the requirements for converting RS/1 tables to IDM relations and IDM relations to RS/1 tables. The following are supported:

integers - short (iINT2) and long (iINT4)
floating point - single (iFLT4) and double precision (iFLT8)
string - null terminated ASCII (iSTRING).

Return codes are supplied at two levels: RS/1 and IDM. RS/1 return codes provide minimal success/failure information and are made directly available as return parameters. IDM return codes must be requested by calling the routine "O$CDUMP" and examining the value in bytes 4-7 as a binary coded integer (int4). This is can also be achieved by calling "O$CURSOR_RPC".

The CDA format and its component fields are described in Appendix A.

1.1 A TYPICAL SEQUENCE OF CALLS

A typical sequence of calls using the interface might appear as in the following figure. The symbol "#" is the RS/1 prompt. Always enclose string constant parameters within single quotes. Quotes embedded within query statements should always be double quotes. Ordinarily it would be advantageous to perform the O$FETCHV loop in a procedure.

```
# SUCCESS = 0
#CALL O$LON['MYUSERNAME','MYPASSWORD']
#ID = O$OPEN[]
#CRC = O$SQL3[ID,'open vino']
#CRC = O$EXEC[ID]
#CRC = O$SQL3[ID,'range of s is stores']
#CRC = O$EXEC[ID]
#CRC = O$SQL3[ID,'delete s where s.storenum = 8']
#CRC = O$EXEC[ID]
#CRC = O$SQL3[ID,'retrieve [s.storenum]']
#CRC = O$EXEC[ID]
#CRC = O$DEFINE_ALL[ID]
#CRC = O$FETCHV[ID,STORENUMBER]
#IF CRC = SUCCESS THEN TYPE STORENUMBER
#IF CRC = SUCCESS THEN CRC = O$FETCHV
#etc. until end of fetch
          .
          .
          .
          .
     or else flush the buffer with
#CALL O$OPT[ID,0,0]
#CRC = O$CLOSE[ID]
#CRC = O$LOGOF[]
#
```

FIGURE 1

2. HIGHER LEVEL ROUTINES

This section describes the higher level routines that perform some of the more common tasks in the IDM-RS/1 interface. You can perform additional tasks by using one or more of the low level interface routines described in Section 4.

RS/1 call ntuples = O$SELECT2TABLE (querystetement,tablename)

Brief Descr. Produces an RS/1 teble from the IDM database, based on e query statement.

where:

ntuples is the variable into which the number of tuples in the resultent RS/1 table will be returned

querystatement is the query stetement which specifies the data which is to be moved into the RS/1 table

tablename is the name of the resultant RS/1 table

Remarks:

This routine produces a table with all the data specified by the query statement. It has the effect of doing an O$OPEN, O$SQL3, O$EXEC, O$DEFINE_ALL, as many O$FETCHT calls as are necessary to retrieve all the data, and an O$CLOSE. If this routine gets any error codes, it prints out the IDM error message (from O$ERMSG) and does an RS/1 error return.

If you omit either argument, RS/1 will prompt you for the required information.

Before calling this routine, you must have set up the IDM interface by using the O$LON call (described in Section 4.).

Example:

```
#N=O$SELECT2TABLE['Select * from EMP','EMP_RS1']
```

RS/1 call REL2TABL[viewname,tablename,database]

Brief Descr. Produces an RS/1 table from an IDM view or relation.

where:

viewname is the name of the IDM view or relation

tablename is the name of the resultant RS/1 table

database is the database in which viewname is to be found

Remarks:

This routine is similer to O$SELECT2TABLE, except it takes an IDM view name or relation name, instead of an query statement, and produces an RS/1 table containing all the data in that view or relation. It opens the designated database, performs an O$DEFINE_ALL and as many O$FETCHT calls as are necessary to copy the relation into the designated RS/1 table. If the database and relation or view selected do not exist, the procedure will exit with an appropriate error message.

If you omit any argument, RS/1 will prompt you for the required information.

Examples:

```
# CALL REL2TABL
Enter relation or view name: mydata
Enter table name: mytab
Enter database name: alldata
Creating table...
Table successfully created.
#
```

RS/1 call TABL2REL(viewname,tablename,database)

Brief Descr. Produces an IDM relation from an RS/1 table.

where:

viewname is the name of the IDM view or relation

tablename is the name of the resultant RS/1 table

database is the database in which the relation is to be created

Remarks:

This routine creates the spacified relation in the selected database using all the data in an RS/1 table. It opens the designated database, creates a relation with one attribute per table column, and appends table data to the relation. Attribute names are determined by the first twelve characters of the column names. This procedure assumes that the RS/1 table selected is in first normal form, that rows containing non-string EMPTY values may be ignored, and that columns containing neither integer, string, or float values may be ignored. Only the first 255 characters of text strings longer then 255 are supportad. These restrictions may be circumvented in particular cases by writing procedures using the lower-level interface routines described in section 4 below.

If you omit any argument, RS/1 will prompt you for the required information.

Example:

```
# CALL REL2TABL
Enter relation or view name: mydata
Enter table name: myteb
Enter database name: alldata
Creating relation...
Relation successfully created.
#
```

3. ROUTINES TO MANAGE THE COMMUNICATIONS AREAS

3.1 ACCESSING CDA AND LDA FIELDS

You can look at some of the fields within a CDA directly through the following IDM-RS/1 calls:

CDA Field	IDM-RS/1 Routine
Return code	crc = O$CURSOR_RCODE(CDA)
Function code	fc = O$CURSOR_FC(CDA)
IDM return code	rc = O$CURSOR_RPC(CDA)

You can access the return code field of an LDA directly by using the call

```
lrc = O$LDA_RCODE()
```

See also O$CDUMP below.

3.2 CHANGING DEFAULTS

There are, by default, a maximum of

- 20 CDAs per program
- 200 fields per query statement (for fetches)
- 10,000 bytes of data which can be transferred by a FETCH operation
- 64 substitution fields per CDA (for BINDS)
- 512 bytes of data per CDA which can be transferred to IDM via O$BINDINFO

For VMS, where dynamic allocation is available, the interface structures are not set up until runtime, and you can change the defaults (only before calling O$LON) by using the following routine:

```
CALL O$SETPARAMS (nCDAs, nfields, fetchsz, nsubst,
bindsz)
```

where

nCDAs	is the new maximum number of CDAs. It is generally advisable to keep this number as small as possible.
nfields	is the new maximum number of fields per query statement
fetchsz	is the new maximum number of bytes of data which can be transferred by a single FETCH operation
nsubst	is the new maximum number of substitution variables per query statement
bindsz	is the new maximum number of bytes of data per CDA which can be transferred to IDM with O$BINDINFO

You can see the current values which are in effect for these parameters by using the call

```
flg = O$GETPARAMNS (nCDAs, nfields, fetchsz, nsubst,
bindsz)
```

The interface will return the current values in the variables nCDAs, nfields and fetchsz. The value returned in flg will be TRUE if the interface is currently active (i.e., if O$LON has been called), otherwise it will be FALSE. Note that all parameters are strictly limited by the maximums set within Britton-Lee VAX runtime support softwsre.

4. BASIC INTERFACE ROUTINES

This section describes the low level IDM-to-RS/1 interface routines that correspond to the routines provided with the IDM system. Each corresponding IDM routine is listed below as "IDM routine" and is described in the IDM PHI document.

Many of the original IDM interface routines, when they take text erguments, can also take a length argument if the text is not null-terminated. For exemple, OSQL3. In all cases on input to IDM, RS/1 will handle the length information itself—you will not specify it. On output from IDM to RS/1, RS/1 will handle the cases as shown in the descriptions below.

RS/1 call: crc = O$BIND (CDA-id, target, target-value)

Brief Descr.: Modifies an query statement after it has been passed to IDM, by assigning a program value to a substitution variable within the statement.

IDM routine irsubst

where

crc is the variable into which the CDA return code will be returned (0 means the O$BIND was successful, non-zero means the call failed).

CDA-id is the variable containing the id of the CDA associated with the query statement containing the substitution variable

target specifies the substitution variable. You can specify it by name or by number. To specify by number, "target" must specify a binary integer. This integer identifies the substitution variable according to its relative position within tha query statement (from left to right, the first substitution variable is number 1, the next is number 2, etc.). To specify by name, "target" ?? must specify the character string name of the substitution variable.

targetvalue is the name of a variable in your program, which contains the value to be substituted into the substitution variable.

Remarks:

After using the O$BIND call, you mey then execute the statement, modify it again using O$BIND, re-execute it, etc. You use O$BIND after an O$SQL or O$SQL3 call and prior to an O$EXEC call. If the same substitution verieble name occurs more than once, a single call to O$BIND will bind them all. If there is more than one substitution verieble name, you must use a separate O$BIND call for each.

RS/1 call: CALL O$CDUMP (CDA-id)

Brief Descr.: Types out the contents of the specified CDA. Bytes 4-7 contain the IDM return code for the most recent command, as a binary coded integer (int4).

IDM routine (none)

where

CDA-id is a variable containing the id of the CDA being examined.

Remarks:

This call is intended for debugging purposes only.

RS/1 call: crc = O$CLOSE (CDA-id)

Brief Descr.: Disconnects a CDA from IDM and frees all resources related to it.

IDM routine irclose

where

crc is the variable into which the CDA return-code will be returned.

CDA-id is the variable containing the id of the CDA to be disconnected.

Remarks

Disconnects the CDA and frees all the resources obtained by the O$OPEN, O$SQL or O$SQL3, and O$EXEC calls which use this CDA.

If the call fails, the reason will be indicated in the IDM CDA return code (see O$CDUMP).

RS/1 call: crc = O$CLOSEALL()

Brief Descr.: Disconnects all open CDAs from IDM and frees all resources related to them.

IDM routine multiple irclose calls

Remarks

Disconnects all the CDAs and frees all the resources obtained by the O$OPEN, O$SQL or O$SQL3, and O$EXEC calls.

The code returned will be non-zero if IDM returned any non-zero values from OCLOSE.

RS/1 call: rc=O$DEFINE (CDA-id,position,exttype,maxlen)

Brief Descr.: Sets up a buffer to receive the data specified in one field of the target list.

IDM routine irbind

where

CDA-id is a variable containing the id of the CDA associated with the query statement.

position is a binary integer which specifies the position of the field in the target list. The fields are numbered left to right, beginning with 1 for the leftmost field.

exttype is a number which specifies the data type to which IDM should convert the data.

maxlen is a binary integer which specifies the maximum length this field will have.

Remarks

Sets up a buffer to receive the data specified in one field of the target list of the query statement associated with "CDA-id".

You issue one O$DEFINE for each field to be retrieved.

RS/1 call: nfields = O$DEFINE_ALL(CDA-id)

Brief Descr.: Sets up buffers to receive the data specified in all the fields in a target list.

where

nfields — is a binary integer which specifies the number of fields (beginning with the leftmost field) in the target list of the query statement that were set up for retrieval by this call.

CDA-id — is a variable containing the id of the CDA associated with the query statement.

Remarks

This routine sets up buffers to receive the data specified in all the fields in a target list. Next, it sets aside internal RS/1 storage for retrieval of all fields and informs the IDM where this storage is and what data types to return. This call does not actually retrieve the data; the FETCH calls do the actual data retrieval. For each field, O$DEFINE_ALL does the equivalent of an O$DESCRIBE (to determine the IDM datatype and other characteristics) and an O$DEFINE (to define a buffer for it).

See also the description of O$DEFINE.

RS/1 call: crc = O$DESCRIBE(CDA-id,pos,maxsize,realsize, rcode,dtype,name,dispsize)

Brief Descr.: Returns the IDM data type and size information for a field in a target list.

where

crc — is the variable into which the CDA return-code will be returned (0 means the O$DESCRIBE was successful; non-zero means the call failed).

CDA-id — is the variable containing the id of the CDA associated with the query statement.

pos — is a binary integer which specifies the position of the field in the target list. The fields and expressions are numbered left to right, beginning with 1 for the leftmost field or expression.

maxsize — returns a binary integer which specifies the maximum size of the field or expression. If the field is defined as iCHAR in the query commands CREATE relation or ALTER relation, then the length returned is the maximum length specified for the

field in that particular CREATE relation or ALTER relation.

realsize — returns a binary integer that indicates the actual size of the data field returned by the last FETCH operation. The value returned is the actual length of the field as stored in the database before it is moved to the user buffer, where padding or truncation may take place. IDM suppresses leading zeros on numeric data and trailing zeros on compressed character data before storing the fields in the database. "Realsize" is valid only if O$DESCRIBE is issued after a FETCH call.

rcode — returns a binary integer indicating the individual field's return code returned by the last FETCH operation. "Rcode" is valid only if O$DESCRIBE is issued after a FETCH call.

dbtype — returns a binary integer that indicates the internal data type of the field as it is stored in the database. Fields stored as ASCII strings return a value of 1; fields stored in IDM extended precision floating-point return a value of 2. Dates return 5; long texts return 8.

name — returns the name of the attribute.

dispsize — returns a binary integer that specifies the maximum display size of the field when it is returned as a character string. "Dispsize" is especially useful when functions are used to modify the representation of a column.

Remarks

O$DESCRIBE operates positionally, one field or expression per call. It references each field or expression in the target list as if each were numbered consecutively, left to right, beginning with 1.

You can use O$DESCRIBE after an O$SQL or O$SQL3, O$EXEC, or before any of the FETCH calls to determine the maximum size, internal datatype, and attribute names of fields to be returned as the result of a query.

If you specify a position number "pos" which is greater than the number of fields in the target list, O$DESCRIBE will return an end-of-file indicator in the return-code. This allows programs to dynamically determine the number of fields to be returned as the result of a query. This is necessary if the program does not know in advance how many fields there are in the target list, as in the case of "SELECT"(SQL) or "RETRIEVE"(IDL).

RS/1 call: msgtxt = O$ERMSG (rcode)

Brief Descr.: Returns an IDM message text

IDM routine errstring [See geterr(3I) in PHI.]

where

msgtxt is the variable into which the error message text will be returned.

rcode specifies the IDM return code for which the message text is to be returned.

Remarks

Returns the IDM error message text corresponding to the IDM return code (from a CDA). RS/1 will pass a 132-byte buffer to the IDM and will return just the text (without trailing blanks).

If there is no message which corresponds with the return code, then the message "Unknown IDM error 'rcode'" is returned.

RS/1 call: crc = O$EXEC (CDA-id)

Brief Descr.: Executes an query statement

IDM routine irexec

where

crc is the variable into which the CDA return code will be returned.

CDA-id is the name of the CDA associated with the query statement being executed.

Remarks

Executes the query statement currently associated with "CDA-id".

If the query statement is a data manipulation, data definition, or data control statement, the entire query function is performed at this time; the "CDA return code" is set. If the query statement is a "SELECT" (SQL) or "RETRIEVE" (IDL), you must explicitly request each tuple of the result using a FETCH call.

RS/1 call: crc = O$FETCHV (CDA-id, val1, val2,...)
or crc = O$FETCHT (CDA-id, tablename, rownumber)
or crc = O$DOFETCH (CDA-id)
or vrc = O$FETCH1VAL (CDA-id, position, vble)

Brief Descr.: Retrieves one or more values from an IDM relation, as specified by the SELECT or RETRIEVE list in an query statement.

O$FETCHV — Retrieves one or more values into variables. This version is useful when you know in advance how many fields there are. You must call O$DEFINE or O$DEFINE_ALL prior to this call.

O$FETCHT — Retrieves values into a row of a table. You must call O$DEFINE or O$DEFINE_ALL prior to this call.

O$DOFETCH — Retrieves (from IDM, into an internal RS/1 buffer) values for all the fields specified by the preceding O$DEFINEs or O$DEFINE_ALLs on the same CDA-id. Operates in conjunction with O$FETCH1VAL.

O$FETCH1VAL — Retrieves one value (from the buffer of values set up by an O$DOFETCH) into a variable. Operates in conjunction with O$DOFETCH. O$FETCH1VAL does not actually call any IDM routine. It only moves values from the RS/1 internal fetch buffer to the user's variable.

where

crc — is the variable into which the CDA return code will be returned.

CDA-id — is the variable containing the id of the CDA associated with the query query statement. The CDA-id in O$FETCH1VAL must be the same as used in the most recent preceding O$DOFETCH.

val1, val2... — are the names of variables in your RS/1 program into which the retrieved values will be placed.

tablename — is the name of the RS/1 table into which the retrieved values are to be placed.

tuplenumber — is the number of the tuple within "tablename" into which the retrieved values are to be placed.

position — is a binary integer which specifies the position of the field in the target list. The fields are

numbered left to right, beginning with 1 for the leftmost field.

vble — is the variable in your RS/1 program into which the retrieved values will be placed.

Remarks

Fields you request in character string format will be left justified and padded with trailing blanks. Character strings that are too long for the field buffer will be truncated and the return code of the CDA will be set to +3. If RS/1 encounters null values on a fetch, the return code of the CDA will bs set to +2 and the buffer will remain unchenged. After the IDM has returned the lest tuple of the query result, any subsequent fetch operations will return an end-of-fetch return code (+4). If the IDM encounters conditions which produce more than one non-zero return code in a single fetch operation, the CDA return code will contain the code of the last condition encountered.

RS/1 call: lrc = O$LOGOF ()

Brief Descr.: Disconnects a program from the IDM interface and frees all IDM resources for the program.

IDM routine — no equiv.

where

lrc — is a variable in which the return code field of the LDA will be returned (0 = no error).

Remarks

Disconnects the current program from IDM and frees all IDM resources owned by this program. Frees up all memory used by the interface.

If the call fails, the reason is indicated in the LDA return code.

O$LOGOF automatically closes any currently open CDAs.

RS/1 call: lrc = O$LON [userid, password[, flag]]

Brief Descr.: Initializes the interface

IDM routine initidmlib

where

lrc is a variable in which the return code field of the LDA will be returned (0=no error).

userid is your IDM user-id.

password is your IDM password.

flag dummy parameter. May be ignored.

Remarks

This call is the initialization routine for the interface. It sets up the communications areas and all other necessary structures (LDA, CDAs, etc.). It returns the return code field of the LDA (the first two bytes of the LDA, which are set to 0 if there is no error, the error return codes are listed in the IDM "Host Software Message Summary".

This call must precede any other IDM interface calls.

A program can execute one and only one O$LON call.

After an O$LON is executed, the program must eventually execute an O$LOGOF call.

RS/1 call: crc = O$NAME (CDA-id, position, name)

Brief Descr.: retrieves the name of an attribute used in a target list of an query statement.

IDM routine irdesc

where

crc is the veriable into which the CDA return-code will be returned.

CDA-id is the variable conteining the id of the CDA associated with the query statement.

position is a binary integer which specifies the position of the attribute in the target list. The fields and expressions are numbered left to right, beginning with 1 for the leftmost field or expression.

name — returns the name of the attribute (associated with the field). If "name" is "0", then the name will not be returned.

Remarks

You can get the name by using the O$DESCRIBE call, as well as by using the O$NAME call.

O$NAME operates positionally, one field or expression per call. It references each field or expression in the target list as if each were numbered consecutively, left to right, beginning with 1.

You can use O$NAME after an O$SQL or O$SQL3 call to get the name of the attribute to be returned.

The maximum length of a attribute name literal expression text is 12 bytes, although the internal buffer allows for up to 64.

If you select a position number "position" which is greater than the number of fields in the RETRIEVE or SELECT list, O$NAME returns an end-of-file indicator in the return-code. As with O$DESCRIBE, this allows programs to dynamically determine the number of fields to be returned as the result of a query.

RS/1 call: CDA-id = O$OPEN ([areasize])

Brief Descr.: Establishes a CDA

IDM routine iropen

where

CDA-id — is the variable into which the CDA id will be returned.

areasize — specifies the size of the query work area (QWA) in IDM (see remarks below).

Remarks

Establishes a CDA for an query statement, as specified by a subsequent O$SQL3 call. RS/1 will manage the CDA, and will choose the CDA-id.

You can determine the return code of the O$OPEN by using the O$CURSOR_RCODE (CDA-id) function with the "CDA-id" as returned by O$OPEN. Generally, if the O$OPEN fails, the returned CDA-id variable will be EMPTY.

The interface allocates one query work area (QWA) for each CDA (i.e., the QWA must be long enough to contain the compiled query statement plus one complete tuple of data from the IDM relation or view being processed). It allocates a seperate QWA for each O$OPEN call. If you do not specify "areasize", then the interface will establish an QWA of the default size. You can override this default by specifying a value for "areasize". If you specify a number between 3 and 32, the interface will interpret this number as being in increments of 1024 bytes (e.g., 3 = 3072 bytes, 4 = 4096 bytes, etc.). If you specify a number from 129 though 32767, the interface will take this number as the number of bytes (e.g., 8003 = 8003 bytes, etc.).

RS/1 call: CRC = O$OPT(CDA-id,roll,wait)

Brief Descr.: Cancels current activity on a specified CDA.

IDM routine irflush, ircancel

where

CDA-id is the current CDA id.

roll dummy integer variable - must be 0

wait dummy integer variable - must be 0

Remarks

This routine cancels any outstanding IDM commands and flushes the buffers.

RS/1 call: crc = O$SQL (CDA-id, query)

Brief Descr.: Passes an SQL query statement to IDM, and associates the statement with an open CDA.

IDM routine irsql

where

crc is the variable into which the CDA return-code will be returned.

CDA-id is the variable containing the id of the CDA to be associated with the query statement.

query is the SQL query statement being passed to IDM. It can be any valid query, data manipulation, data definition, or data control statement.

Remarks

This function causes RS/1 to clear its internal information about fields that may have been used for a previous query statement for this CDA.

IDM checks the query statement for validity, and returns an error code in "crc" (0 means the call executed successfully; non-zero means the call failed). The error return codes are listed in the IDM "Host Software Message Summary". You can use O$ERMSG to get the error text associated with an IDM error code.

"querystatement" may contain substitution variables anywhere a constant is permitted (substitution variables are identified by preceding the variable name with a colon--e.g., :TRIALNO). If the query statement has substitution variables, you must use the O$BIND call to bind in values for the substitution variables. You use the O$EXEC call to execute the statement before retrieving values.

RS/1 call: crc = O$SQL3 (CDA-id, query)

Brief Descr.: Passes an IDL query statement to IDM, and associates the statement with an open CDA.

IDM routine iridl

where

crc is the variable into which the CDA return-code will be returned.

CDA-id is the variable containing the id of the CDA to be associated with the query statement.

query is the idl statement being passed to IDM. It can be any valid query, data manipulation, data definition, or data control statement.

Remarks

This function causes RS/1 to clear its internal information about fields that may have been used for a previous query statement for this CDA.

IDM checks the query statement for validity, and returns an error code in "crc" (0 means the call executed successfully; non-zero means the call failed). The error return codes are listed in the IDM "Host Software Message Summary". You can use O$ERMSG to get the error text associated with an IDM error code.

"querystatement" may contain substitution variables anywhere a constant is permitted (substitution variables are identified by preceding the variable name with a colon--e.g., :TRIALNO). If the query statement has substitution variables, you must use the O$BIND call to bind in values for the substitution variables. You use the O$EXEC call to execute the statement before retrieving values.

Appendix A: CDA DATA AREA

This appendix describes the CDA and its component fields.

The CDA is a 64-byte data area defined within a user program. The CDA contains status information on an active query operation. There is one CDA data area for each active query statement within a user program.

Each CDA has a CDA-id, which RS/1 assigns to it with an O$OPEN call. You refer to an query statement by specifying the CDA-id of the CDA associated with that query statement.

The CDA format is as follows:

Byte Numbers	Field Name	Field Description
0-1	Return code	A binary number that indicates the completion code for the specified query operation. Zero indicates a successful result. A positive return code indicates a successful result with an exceptional condition. A negative return code indicates that an error was encountered in attempting to perform the specified operation. See IDM documentation for a complete list of return codes.
2-3	Function Type	(Used internally in IDM, not supported for user environments.)
4-7	Return Process Code	A binary number that indicates the code returned by the IDM for each call processed. Use this code for calls to O$ERMSG.
8-9	Parse Error	A binary number indiceting the offset (in chereCters) into the query text where the parse error occurred. This field is valid only efter an O$SQL3 call.
10	Function Code	An operation code indicating the type of IDM function requested. The function codes are: 02 OSQL iridl 04 OEXEC irexec 06 OBIND irsubst 08 ODFINN irbind 10 ODSRBN 12 OFETCH irfetch 14 OOPEN iropen

	16	OCLOSE irclose
	22	ODSC irdesc
	24	ONAME irdesc
	26	OSQL3 irsql
	50	OBINDN irsubst
11	(filler)	
12-63	IDM System Peremeter Area	(Used internally, not supported for user environments.)

APPENDIX B: CODES

1. DATA TYPES

This interface will support only the following data types:
two and four byte integer
four and eight byte floating point
string.
All other data types will generate an error when fetched.

The corresponding codes for these data types are:

I2 -
I4 -
F4 -
F8 -
A -

2. CDA RETURN CODES

The following are the return code values for the CDA and their meanings:

The following are the return code values for the LDA and their meanings:

See PHI for IDM return code values and their meanings.

In general, 0 represents success, positive non-zero values a warning, and negative values are fetal errors.