Calling Stored Procedures

Stored procedures allow you to directly invoke a procedure located on the database using the Adabas direct command PC.

This chapter covers the following topics:

- Stored Procedure Link Routine (STPLNKnn)
- Setting Up the PC Command
- Examples

Stored Procedure Link Routine (STPLNKnn)

The PC command is used in conjunction with the stored procedure link routine STPLNKnn to invoke a stored procedure.

STPLNKnn is provided in the SYSSPT library in source format.

The examples STPLNK01, STPLNK02, and STPLNK03 from the library SYSSPT illustrate the use of the PC command in calling stored procedures. Each example passes parameters to the routine in a different way.

You may use these examples or write your own routines. If you use the examples, you may change the routine code or name to meet standards or requirements at your site. You may choose to include the routine name as inline code in the main Natural program.

In the three examples, the PC command is invoked by calling the Natural routine CMADA. If you do not want to code this entry name directly, you can issue a CALLNAT to the Natural subprogram USR1043N in library SYSEXT instead. The advantage of using the CALLNAT alternative is to insulate your code from changes to the name "CMADA" that may occur across time or across platforms.

Setting Up the PC Command

The Adabas control block (ACB) for the PC command (direct call) must be set up before STPLNKnn is used to invoke a stored procedure request.

This section covers the following topics:

- PC Command Function and Use
- ACB Interface Direct Call Control Block and Buffer Overview
- ACBX Interface Direct Call Control Block and Buffer Overview
- Buffers

PC Command Function and Use

The PC command provides a mechanism for invoking stored procedures.

Parameters are passed using the record buffer; they are subsequently updated by the stored procedure and returned to the caller.

The format buffer may be used to define the parameters to the procedure. Such information may be relevant when calling the record buffer extraction routine.

ACB Interface Direct Call Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	Not used	Not used	Not used
Command Code	3-4	alphanumeric	F	U
Command ID	5-8	alphanumeric	F	U
File Number	9-10	alphanumeric	F	U
Response Code	11-12	binary	Not used	A
	13-24	Not used	Not used	Not used
Format Buffer Length	25-26	binary	F	U
Record Buffer Length	27-28	binary	F	U
	29-36	Not used	Not used	Not used
Additions 1	37-44	alphanumeric	F	U
Additions 2	45-48	binary / binary		A
Additions 3	49-56	alphanumeric	F	A
Additions 4	57-64	alphanumeric	Not used	A
	65-76	Not used	Not used	Not used
User Area	77-80			U

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	F	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call

Control Block Field Descriptions

Command Code (ACBCMD)

PC

Command ID (ACBCID)

Set this field to the value ’STPx’ where "x" is any value.

File Number (ACBFNR)

By default, indicates the trigger file database ID and file number.

For a one-byte database ID, set CB-DBID; for a two-byte database ID, set CB-RSP with CB-CALL-TYPE set to X’30’.

You may specify the file number of any other user file in conjunction with the format buffer. File number should be consistent with the format buffer so that the record buffer extraction (STPRBE) routine may be used to interpret or retrieve field values according to the file-field definitions.

Specify the binary number of the file to be read in this field. For the physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B’0000 0000’).
- For a two-byte file number, use both bytes (9 and 10) of the field.

  Note:
  When using two-byte file numbers and database IDs, a X’30’ must be coded in the first byte of the control block.

Response Code (ACBRSP)

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the rightmost half of the Additions 2 field, are described in the *Adabas Messages and Codes* documentation.

Format Buffer Length (ACBFBL)

The format buffer length (in bytes). The actual format buffer area defined in the user program must be at least as large as the length specified.

Record Buffer Length (ACBRBL)

The record buffer length (in bytes). The actual record buffer area defined in the user program must be at least as large as the length specified.

Additions 1 - Name of the Stored Procedure - (ACBADD1)

The name of the stored procedure.

Additions 2 - Length of Compressed and Decompressed Record - (ACBADD2)

The PC command returns a response from the procedure executed in bytes 1 and 2 of this field.

Additions 3 - Stored Procedure Options- (ACBADD3)

This field indicates the options to be used when the stored procedure request is issued:

Byte 1 is Type:	A=asynchronous, P=participating, N=non-participating
Byte 2 is Parm:	N=none, C=control, E=error/response, X=control with ACBX
Byte 3 is RecB:	N=none, A=access, U=update

Additions 4 (ACBADD4)

The PC command returns a response from the procedure executed in bytes 1 and 2 of this field. Bytes 3 and 4 are set to X’0011 (17) to indicate "stored procedure".

ACBX Interface Direct Call Control Block and Buffer Overview

Control Block

Field	Position	Format	Before Adabas Call	After Adabas Call
	1-2	binary	---	---
Version Indicator	3-4	binary	F	
	5-6	binary	---	---
Command Code	7-8	alphanumeric	F	U
	9-10	binary	---	---
Response Code	11-12	binary	---	A
Command ID	13-16	alphanumeric/ binary	F	U
Database ID	17-20	numeric	F	U
File Number	21-24	numeric	F	U
	25-56	---	---	---
Additions 1	57-64	alphanumeric	F	U
Additions 2	65-68	binary	---	A
Additions 3	69-76	alphanumeric	F	A
Additions 4	77-84	alphanumeric	---	A
	85-114	---	---	---
Error Subcode	115-116	binary	---	A
	117-144	---	---	---
Command Time	145-152	binary	---	A
User Area	153-168	not applicable	---	U
	169-193	---	---	---

Buffer Areas

Buffer	Before Adabas Call	After Adabas Call
Format	F	U
Record	F	A

where:

F	Supplied by user before Adabas call
A	Supplied by Adabas
U	Unchanged after Adabas call

Control Block Field Descriptions

Version Indicator (ACBXVER)

F2

Command Code (ACBXCMD)

A1

Response Code (ACBXRSP)

Adabas returns the response code for the command in this field. Response code 0 indicates that the command was executed successfully. Non-zero response codes, which can also have accompanying subcodes returned in the Error Subcode (ACBXERRC) field, are described in the *Adabas Messages and Codes Manual* documentation.

Command ID (ACBXCID)

Set this field to the value STP*x* where *x* is any value.

Database ID (ACBXDBID)

Specify the database ID for a call.

If this field is set to binary zeros, the Adabas API uses either the database ID from the ADARUN cards provided in DDCARD input data, or the default database ID value provided in the LNKGBLS module linked with or loaded by the link routine.

File Number (ACBXFNR)

By default, indicates the trigger file file number.

You may specify the file number of any other user file in conjunction with the format buffer. File number should be consistent with the format buffer so that the record buffer extraction (STPRBE) routine may be used to interpret or retrieve field values according to the file-field definitions.

Specify the binary number of the file to be read in this field. For the physical direct calls, specify the file number as follows:

- For a one-byte file number, enter the file number in the rightmost byte (10); the leftmost byte (9), should be set to binary zero (B’0000 0000’).
- For a two-byte file number, use both bytes (9 and 10) of the field.

Additions 1 - Name of the Stored Procedure - (ACBXADD1)

The name of the stored procedure.

Additions 2 - Length of Compressed and Decompressed Record - (ACBXADD2)

If the command is processed successfully, the following information is returned in this field:

- If the record buffer contains at least one valid field value, the leftmost two bytes contain the length (in binary form) of the compressed record accessed;
- The rightmost two bytes contain the length (in binary form) of the decompressed fields selected by the format buffer and accessed.

Note:
This length information is not returned when the prefetch feature is being used.

Additions 3 - Stored Procedure Options- (ACBXADD3)

This field indicates the options to be used when the stored procedure request is issued:

Byte 1 is Type:	A=asynchronous, P=participating, N=non-participating
Byte 2 is Parm:	N=none, C=control, E=error/response, X=control with ACBX
Byte 3 is RecB:	N=none, A=access, U=update

Additions 4 (ACBXADD4)

The PC command returns a response from the procedure executed in bytes 1 and 2 of this field. Bytes 3 and 4 are set to X’0011 (17) to indicate "stored procedure".

Error Subcode (ACBXERRC)

If the command returns a nonzero response code, this field contains a subcode defining the exact response code meaning. Response codes and their subcodes are defined in the *Adabas Messages and Codes Manual* documentation.

Buffers

The following buffers apply to the PC command:

- Format Buffer
- Record Buffer

Format Buffer

The user specifies the fields to be read in this buffer. Format buffer syntax and examples are provided in the *Adabas Command Reference* documentation.

The format buffer may optionally be used to convey the definition of the parameters being passed in the record buffer. The syntax must be consistent with that of a format buffer for a normal command, or be set to "." if it is not to be used.

The field names used in the format buffer should normally be meaningful so that the stored procedure can acquire the values of each parameter from the record buffer extraction (STPRBE) routine (see *Record Buffer Extraction Routine (STPRBE)*). Length must be used if the stored procedure routine does not provide one. Alternatively, if the field names correspond to the actual file number specified in the ACB, then the STPRBE routine will be able to determine the length of the field/parameter.

When issuing stored procedures across platforms, it is essential to also specify the field type of each parameter; i.e., "A" - alphanumeric, "B" - binary, "U" - unpacked, etc.

See *Format Buffer* for more information.

Record Buffer

Adabas returns the requested field values in this buffer. All values are returned according to the standard format and length of the field unless the user specifies a different length and/or format in the format buffer.

The record buffer is available for passing any parameters from the caller to the stored procedure and/or from the stored procedure to the caller. The layout or DSECT of the record buffer must be coordinated between the caller and the actual stored procedure routine itself.

The record buffer is available for participating and non-participating (sync) type requests *only* by using the record buffer extraction (STPRBE) routine. See *Record Buffer Extraction Routine (STPRBE)*.

Whether the record buffer is used for access or update is specified by the caller using the Additions 4 field.

See *Using the Format and Record Buffers* for more information

Examples

This section contains the example programs and data areas listed in the following table. Source code is provided during the installation procedure and is located in the library SYSSPT.

Name	Description
STPLNK01	This stored procedure link routine passes parameters as fixed length and fixed number.
STPLNK02	In this stored procedure link routine, a maximum of five parameters may be passed to the procedure; the length of each parameter is contained in the first two bytes of the parameter.
STPLNK03	Like STPLNK03, a maximum of five parameters may be passed to the procedure; however, the length of each parameter is contained in a preceding, separate parameter.

STPLNK01

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *  Subprogram : STORPROC/STPLNK01
0040 *  Author     : Adabas Development
0050 *
0060 *  Function   : Sample Routine 01 to invoke a stored procedure
0070 *               This example expects fixed parameter definitions
0080 *  Remarks    : This routine will set up the buffers and issue the call
0090 *               to invoke a stored procedure routine directly.
0100 *               Once processing is completed, control is returned to
0110 *               the caller.
0120 *               Parameter RESP must be set to zero if processing is
```

```
0130 *               successful.
0140 *
0150 *  Parameters : The following fields in the ACB must be set up to invoke
0160 *               the stored procedure request.
0170 *
0180 *               Command Code: 'PC'
0190 *               Command ID  : 'STPx' - where x is any value
0200 *               Database ID : Database of the respective trigger file
0210 *                             Set CB-DBID for a one byte DBID
0220 *                             Set CB-RSP  for a two byte DBID with
0230 *                               CB-CALL-TYPE set to H'30'
0240 *               File Number : Set to the trigger file number of the
0250 *                             target database (normal one-byte versus
0260 *                             two-byte FNRs is applicable) by default
0270 *                             or any other file used in conjunction with
0280 *                             the format buffer.
0290 *               FB Length   : Length of the format buffer
0300 *               RB Length   : Length of the record buffer
0310 *               Additions 1 : Name of the stored procedure
0320 *               Additions 3 :
0330 *                  Byte 1   : Type ("A"sync, "P"art, "N"on-Partic)
0340 *                  Byte 2   : Parm ("N"one, "C"ntl, "E"rror/Resp)
0350 *                  Byte 3   : RecB ("N"one, "A"ccess, "U"pdate)
0360 *
0370 *
0380 *  Format Buff: The format buffer is an optional buffer that may be used
0390 *               to convey the definition of the parameters being passed
0400 *               in the record buffer. The syntax must be consistent with
0410 *               that of a format buffer for a normal command, or be set
0420 *               to "." if it not to be used.
0430 *
0440 *               The field names used in the format buffer should
0450 *               normally be meaningful so that the stored procedure can
0460 *               get the values of each parameter from the record buffer
0470 *               extraction (STPRBE) routine. Length must be used if the
0480 *               stored procedure routine does not provide one.
0490 *               Alternatively, if the field names correspond to the
0500 *               actual file number specified in the ACB, then the STPRBE
0510 *               routine will be able to determine the length of the
0520 *               field/parameter.
0530 *
0540 *               When issuing stored procedures across platforms, it is
0550 *               essential to also specify the field type of each
0560 *               parameter; i.e., "A" - alphanumeric, "B" - binary, "U"
0570 *               - unpacked etc.
0580 *
0590 *
0600 *  Record Buff: The record buffer is available for passing any
0610 *               parameters from the caller to the stored procedure
0620 *               and(or) from the stored procedure to the caller.
0630 *               The layout/DSECT of the record buffer must be
0640 *               coordinated between the caller and the actual stored
0650 *               procedure routine itself.
0660 *
0670 *               The record buffer is available for participating
0680 *               and non-participating (sync) type requests via the
0690 *               the record buffer extraction (STPRBE) routine, only.
0700 *
0710 *               Determination of the record buffer being for access or
0720 *               update is specified by the caller via the additions 3
0730 *               field (see above).
0740 *
```

```
0750 ************************************************************************
0760 DEFINE DATA PARAMETER
0770  01  REQ-TYPE  (A1)              /* Optional request ID type
0780  01  P-PROC    (A8)              /* Procedure name
0790  01  P-PARM1 (A100)              /* Single parameter
0800  01  P-MSG    (A72)             /* Message corresponding to the RESP
0810  01  RESP      (N4)              /* Response code of proc. request
0820            LOCAL USING STPLCB
0830            LOCAL
0840  01  FB    (A16) INIT<'AA,100,A.'>
0850  01  ET-CNT         (P3)
0860 END-DEFINE
0870 FORMAT PS=0
0880 *
0890 RESET CB                        /* Clear the ACB
0900 MOVE 'STP'   TO CB-CID          /* Command ID
0910 MOVE 'PC'    TO CB-CMD          /* Command code
0920 MOVE 222     TO CB-DBID         /* Database ID
0930 MOVE 12      TO CB-FNR          /* Default to TRG file number
0940 MOVE 9       TO CB-FBL          /* FB length
0950 MOVE 100     TO CB-RBL          /* RB length
0960 IF P-PROC = ' '                 /* Did we get a procedure name?
0970   DO
0980     MOVE 1 TO RESP
0990     MOVE 'Invalid Procedure Name specified' TO P-MSG
1000     ESCAPE ROUTINE
1010   DOEND
1020 MOVE P-PROC     TO CB-ADD1      /* Stored procedure name
1030 MOVE 'NCA     ' TO CB-ADD3      /* Options: N - Sync (non-partic)
1040 *                               /*          C - Control parms
1050 *                               /*          A - RecBuff for access
1060 *
1070 CALL 'CMADA' USING CB FB P-PARM1 /* Invoke the stored procedure
1080 *
1090 MOVE CB-RSP TO RESP
1100 MOVE 'Check Response code returned for this request' TO P-MSG
1110 * PRINT (CD=YE) 'Resp ..' (YEI) CB-RSP(EM=HH) 'Add2' CB-ADD2(EM=H(8))
1120 *              'Add4' CB-ADD4(EM=H(8))
1130 *
1140 END
```

STPLNK02

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *  Subprogram : STORPROC/STPLNK02
0040 *  Author     : Adabas Development
0050 *
0060 *  Function   : Sample routine 02 to invoke a stored procedure
0070 *               This example expects up to 5 different variable-length
0080 *               parameters. The length of each parameter is specified
0090 *               as the first two bytes of each parameter. Length is
0100 *               inclusive of the two-byte length itself.
0110 *  Remarks    : This routine will set up the buffers and issue the call
0120 *               to invoke a stored procedure routine directly.
0130 *               Once processing is completed, control is returned to
0140 *               the caller.
0150 *               Parameter RESP must be set to zero if processing is
0160 *               successful.
0170 *
0180 *  Parameters : The following fields in the ACB must be set up to invoke
0190 *               the stored procedure request.
```

```
0200 *
0210 *               Command Code: 'PC'
0220 *               Command ID  : 'STPx' - where x is any value
0230 *               Database ID : Database of the respective trigger file
0240 *                             Set CB-DBID for a one-byte DBID
0250 *                             Set CB-RSP  for a two-byte DBID with
0260 *                               CB-CALL-TYPE set to H'30'
0270 *               File Number : Set to the trigger file number of the
0280 *                             target database (normal one-byte versus
0290 *                             two-byte FNRs is applicable) by default
0300 *                             or any other file used in conjunction with
0310 *                             the format buffer.
0320 *               FB Length   : Length of the format buffer
0330 *               RB Length   : Length of the record buffer
0340 *               Additions 1 : Name of the stored procedure
0350 *               Additions 3 :
0360 *                  Byte 1   : Type ("A"sync, "P"art, "N"on-Partic)
0370 *                  Byte 2   : Parm ("N"one, "C"ntl, "E"rror/Resp)
0380 *                  Byte 3   : RecB ("N"one, "A"ccess, "U"pdate)
0390 *
0400 *
0410 *  Format Buff: The format buffer is an optional buffer that may be used
0420 *               to convey the definition of the parameters be ingpassed
0430 *               in the record buffer. The syntax must be consistent with
0440 *               that of a format buffer for a normal command, or be set
0450 *               to "." if it not to be used.
0460 *
0470 *               The field names used in the format buffer should
0480 *               normally be meaningful so that the stored procedure can
0490 *               get the values of each parameter via the record buffer
0500 *               extraction (STPRBE) routine. Length must be used if the
0510 *               stored procedure routine does not provide one.
0520 *               Alternatively, if the field names correspond to the
0530 *               actual file number specified in the ACB, then the STPRBE
0540 *               routine will be able to determine the length of the
0550 *               field/parameter.
0560 *
0570 *               When issuing stored procedures across platforms, it is
0580 *               essential to also specify the field type of each
0590 *               parameter; i.e., "A" - alphanumeric, "B" - binary, "U"
0600 *               - unpacked etc.
0610 *
0620 *
0630 *  Record Buff: The record buffer is available for passing any
0640 *               parameters from the caller to the stored procedure
0650 *               and(or) from the stored procedure to the caller.
0660 *               The layout/DSECT of the record buffer must be
0670 *               coordinated between the caller and the actual stored
0680 *               procedure routine itself.
0690 *
0700 *               The record buffer is available for participating
0710 *               and non-participating (sync) type requests via the
0720 *               the record buffer extraction (STPRBE) routine, only.
0730 *
0740 *               Determination of the record buffer being for access or
0750 *               update is specified by the caller via the additions 3
0760 *               field (see above).
0770 *
0780 ************************************************************************
0790 DEFINE DATA PARAMETER
0800  01  REQ-TYPE   (A1)
0810  01  P-PROC     (A8)              /* Procedure name
```

```
0820  01  P-OPTIONS  (A8)
0830  01  REDEFINE P-OPTIONS
0840    02  P-TYPE   (A1)             /* Async versus sync procedure
0850    02  P-PARMS  (A1)             /* Parm type for procedure
0860    02  P-RECB   (A1)             /* Rec buffer access
0870  01  P-PARM1(A1/1:V)             /* Variable-length parameter
0880 *                                    first 2 bytes set to incl. length
0890  01  P-PARM2(A1/1:V)             /* Variable-length parameter 2
0900  01  P-PARM3(A1/1:V)             /* Variable-length parameter 3
0910  01  P-PARM4(A1/1:V)             /* Variable-length parameter 4
0920  01  P-PARM5(A1/1:V)             /* Variable-length parameter 5
0930  01  P-MSG     (A72)             /* Message corresponging to the RESP
0940  01  RESP       (N4)             /* Response code of proc request
0950            LOCAL USING STPLCB
0960            LOCAL
0970  01  SUB        (I2)
0980  01  SUB1       (I2)
0990  01  SUB2       (I2)
1000  01  SUB3       (I2)
1010  01  SUB4       (I2)
1020  01  FB        (A48)
1030  01  REDEFINE FB
1040    02  FB-FIELD  (8)
1050      03  FB-FLD (A3)
1060      03  FB-LEN (N3)
1070  01  RB    (A1/1000)            /* Max length for all parms
1080  01  W-ADD3     (A8)
1090  01  REDEFINE W-ADD3
1100    02  W-TYPE   (A1)
1110    02  W-PARMS  (A1)
1120    02  W-RECB   (A1)
1130  01  #LENGTH    (B2)
1140  01  REDEFINE #LENGTH
1150    02  #LENG  (A1/2)
1160  01  W-LENG   (P5/5)
1170 END-DEFINE
1180 FORMAT PS=0
1190 *
1200 * In this example, we will say that each parameter has an individual
1210 * maximum length of 200; however, the limit may be established as a
1220 * total of all parameters. Since our max. record buffer is 1000 then the
1230 * maximum of all parameters cannot exceed 1000. This may be changed as
1240 * required by the user.
1250 *
1260 FOR SUB1 1 5                      /* Get all the parameter lengths
1270   DECIDE ON FIRST VALUE OF SUB1
1280     VALUE 1  MOVE P-PARM1(1:2) TO #LENG(1:2) /* Get Parm1 length
1290              IF #LENGTH < 3 /* Min length with inclusive length
1300                DO
1310                  MOVE 16 TO RESP
1320                  MOVE 'Invalid Length for Parameter 1. Must be 3-200'
1330                       TO P-MSG
1340                  ESCAPE ROUTINE
1350                DOEND
1360     VALUE 2  MOVE P-PARM2(1:2) TO #LENG(1:2) /* Get Parm2 length
1370     VALUE 3  MOVE P-PARM3(1:2) TO #LENG(1:2) /* Get Parm3 length
1380    VALUE 4  MOVE P-PARM5(1:2) TO #LENG(1:2) /* Get Parm4 length
1390    VALUE 5  MOVE P-PARM1(1:2) TO #LENG(1:2) /* Get Parm5 length
1400    ANY      IF #LENGTH = H'4040'  /* Is length Blanks?
1410               RESET #LENGTH       /*   yes, then treat as dummy parm
1420             MOVE #LENGTH TO W-LENG(SUB1)
1430             IF W-LENG(SUB1) > 202 /* For our example, we limit the length
```

```
1440               DO
1450                 MOVE 4 TO RESP
1460                 MOVE SUB1 TO FB-LEN(SUB1)
1470                 COMPRESS 'Invalid Length for Parameter' FB-LEN(SUB1)
1480                          '. Max is 200.' INTO P-MSG
1490                 ESCAPE ROUTINE    /* Terminate processing with error
1500               DOEND
1510             SUBTRACT 2 FROM W-LENG(SUB1) /* ACTUAL parm length
1520    NONE     IGNORE
1530  END-DECIDE
1540 CLOSE LOOP (1260)
1550 *
1560 IF P-PROC = ' '                    /* Did we get a procedure name?
1570   DO
1580     MOVE 1 TO RESP
1590     MOVE 'Invalid Procedure Name specified' TO P-MSG
1600     ESCAPE ROUTINE
1610   DOEND
1620 IF NOT (P-TYPE = 'A' OR= 'N' OR= 'P' OR= ' ')
1630   DO                               /* Async, participating, non-partic.
1640     MOVE 2 TO RESP
1650     MOVE 'Proc Type must be A, N, P or " "' TO P-MSG
1660     ESCAPE ROUTINE
1670   DOEND
1680 IF NOT (P-PARMS = 'C' OR= 'E' OR= 'N' OR= ' ')
1690   DO                               /* Cntrl, Error/Resp, None
1700     MOVE 3 TO RESP
1710     MOVE 'Parameter Type must be C, E, N or " "' TO P-MSG
1720     ESCAPE ROUTINE
1730   DOEND
1740 IF NOT (P-RECB = 'A' OR= 'N' OR= 'U' OR= ' ')
1750   DO                               /* Access, None, Update
1760     MOVE 3 TO RESP
1770     MOVE 'Parameter access must be Access, None or Update' TO P-MSG
1780     ESCAPE ROUTINE
1790   DOEND
1800 *
1810 * Next we merge all the passed parameters into a single contiguous
1820 * buffer which will be used as the record buffer for the call. The
1830 * format buffer will also be set up to indicate the 'structure' of the
1840 * record buffer for use by the invoked procedure.
1850 *
1860 MOVE 1 TO SUB
1870 *
1880 FOR SUB3 1 5                      /* Step through all parameters
1890   IF W-LENG(SUB3) < 3             /* Check min. length of a parameter
1900     DO
1910       MOVE '.' TO FB-FLD(SUB3)
1920       ESCAPE BOTTOM               /* None, so assume we have all parms
1930     DOEND
1940   MOVE W-LENG(SUB3) TO SUB1
1950   ADD SUB1 TO SUB2
1960   DECIDE ON FIRST VALUE OF SUB1    /* Move parms into the RB
1970     VALUE 1 MOVE 'P1,' TO FB-FLD(1)
1980             MOVE P-PARM1 (3:SUB1) TO RB(SUB:SUB2)
1990     VALUE 2 MOVE 'P2,' TO FB-FLD(2)
2000             MOVE P-PARM2 (3:SUB1) TO RB(SUB:SUB2)
2010     VALUE 3 MOVE 'P3,' TO FB-FLD(3)
2020             MOVE P-PARM3 (3:SUB1) TO RB(SUB:SUB2)
2030     VALUE 4 MOVE 'P4,' TO FB-FLD(4)
2040             MOVE P-PARM4 (3:SUB1) TO RB(SUB:SUB2)
2050     VALUE 5 MOVE 'P5,' TO FB-FLD(5)
```

```
2060             MOVE P-PARM5 (3:SUB1) TO RB(SUB:SUB2)
2070     ANY     ADD SUB1 TO SUB
2080             MOVE SUB1 TO FB-LEN(SUB3)
2090     NONE    IGNORE
2100   END-DECIDE
2110 *
2120 CLOSE LOOP (1880)
2130 *
2140 * Now we start setting up the CB and do some additional validation.
2150 * When moving in the procedure options, we allow for defaults.
2160 *
2170 RESET CB                          /* Clear the ACB
2180 MOVE 'STP'   TO CB-CID            /* Command ID
2190 MOVE 'PC'    TO CB-CMD            /* Command code
2200 MOVE 77      TO CB-DBID           /* Database ID
2210 MOVE 22      TO CB-FNR            /* File number
2220 MOVE 48      TO CB-FBL            /* FB length
2230 MOVE 1000    TO CB-RBL            /* RB length
2240 MOVE P-PROC  TO CB-ADD1           /* Stored procedure name
2250 *
2260 MOVE 'A' TO W-TYPE                 /* Set the default options
2270 MOVE 'C' TO W-PARMS
2280 MOVE 'N' TO W-RECB
2290 IF NOT (P-TYPE = ' ')             /* Should we default to Async?
2300   MOVE P-TYPE TO W-TYPE
2310 IF NOT (P-PARMS = ' ')            /* Should we default to Contrl?
2320   MOVE P-PARMS TO W-PARMS
2330 IF NOT (P-RECB = ' ')             /* Should we default to None?
2340   MOVE P-RECB TO W-RECB
2350 MOVE W-ADD3  TO CB-ADD3           /* Options for request
2360 *
2370 CALL 'CMADA' USING CB FB RB(1)   /* Invoke the stored procedure
2380 *
2390 IF CB-RSP NE 0
2400   DO
2410     PRINT (CD=YE) 'Resp ..' (YEI) CB-RSP(EM=HH) 'Add2' CB-ADD2(EM=H(4))
2420                   'Add3' CB-ADD3(EM=H(8)) 'Add4' CB-ADD4(EM=H(8))
2430     ESCAPE ROUTINE
2440   DOEND
2450 *
2460 * Now we need to restore the parameters back into the user's area,
2470 * in case the data was modified. This can happen only if the record
2480 * buffer was modifiable; i.e., P-RECB was set to 'U'.
2490 *
2500 IF  CB-RSP = 0                    /* Was everything okay
2510 AND P-RECB = 'U'                  /* Update: Parms may have been updated
2520   DO
2530     MOVE 1 TO SUB
2540     RESET SUB2
2550     FOR SUB1 1 5
2560       ADD  W-LENG(SUB1) TO SUB2
2570       MOVE W-LENG(SUB1) TO SUB3
2580       DECIDE ON FIRST VALUE OF SUB1 /* Restore parm from RB
2590         VALUE 1   ASSIGN P-PARM1 (3:SUB3) = RB(SUB:SUB2)
2600         VALUE 2   ASSIGN P-PARM2 (3:SUB3) = RB(SUB:SUB2)
2610         VALUE 3   ASSIGN P-PARM3 (3:SUB3) = RB(SUB:SUB2)
2620         VALUE 4   ASSIGN P-PARM4 (3:SUB3) = RB(SUB:SUB2)
2630         VALUE 5   ASSIGN P-PARM5 (3:SUB3) = RB(SUB:SUB2)
2640         ANY       ADD W-LENG(SUB1) TO SUB /* Get next position
2650         NONE      IGNORE
2660       END-DECIDE
```

```
2670     CLOSE LOOP(2550)
2680   DOEND
2690 *
2700 END
```

STPLNK03

```
0010 ************************************************************************
0020 *  Application: Adabas Stored Procedures
0030 *  Subprogram : STORPROC/STPLNK03
0040 *  Author     : Adabas Development
0050 *
0060 *  Function   : Sample routine 03 to invoke a stored procedure
0070 *               This example expects up to five different variable-length
0080 *               parameters. Parameter lengths are passed as extra
0090 *               parameters.
0100 *  Remarks    : This routine will set up the buffers and issue the call
0110 *               to invoke a stored procedure routine directly.
0120 *               Once processing is completed, control is returned to
0130 *               the caller.
0140 *               Parameter RESP must be set to zero if processing is
0150 *               successful.
0160 *
0170 *  Parameters : The following fields in the ACB must be set up to invoke
0180 *               the stored procedure request.
0190 *
0200 *               Command Code: 'PC'
0210 *               Command ID  : 'STPx' - where x is any value
0220 *               Database ID : Database of the respective trigger file
0230 *                             Set CB-DBID for a one-byte DBID
0240 *                             Set CB-RSP  for a two-byte DBID with
0250 *                               CB-CALL-TYPE set to H'30'
0260 *               File Number : Set to the trigger file number of the
0270 *                             target database (normal one-byte versus
0280 *                             two-byte FNRs is applicable) by default
0290 *                             or any other file used in conjunction with
0300 *                             the format buffer.
0310 *               FB Length   : Length of the format buffer
0320 *               RB Length   : Length of the record buffer
0330 *               Additions 1 : Name of the stored procedure
0340 *               Additions 3 :
0350 *                  Byte 1   : Type ("A"sync, "P"art, "N"on-Partic)
0360 *                  Byte 2   : Parm ("N"one, "C"ntl, "E"rror/Resp)
0370 *                  Byte 3   : RecB ("N"one, "A"ccess, "U"pdate)
0380 *
0390 *
0400 *  Format Buff: The format buffer is an optional buffer that may be used
0410 *               to convey the definition of the parameters being passed
0420 *               in the record buffer. The syntax must be consistent with
0430 *               that of a format buffer for a normal command, or be set
0440 *               to "." if it not to be used.
0450 *
0460 *               The field names used in the format buffer should
0470 *               normally be meaningful so that the stored procedure can
0480 *               otain the values of each parameter via the record buffer
0490 *               extraction (STPRBE) routine. Length must be used if the
0500 *               stored procedure routine does not provide one.
0510 *               Alternatively, if the field names correspond to the
0520 *               actual file number specified in the ACB, then the STPRBE
0530 *               routine will be able to determine the length of the
0540 *               field/parameter.
0550 *
```

```
0560 *               When issuing stored procedures across platforms, it is
0570 *               essential to also specify the field type of each
0580 *               parameter; i.e., "A" - alphanumeric, "B" - binary, "U"
0590 *               - unpacked etc.
0600 *
0610 *
0620 *  Record Buff: The record buffer is available for passing any
0630 *               parameters from the caller to the stored procedure
0640 *               and(or) from the stored procedure to the caller.
0650 *               The layout/DSECT of the record buffer must be
0660 *               coordinated between the caller and the actual stored
0670 *               procedure routine itself.
0680 *
0690 *               The record buffer will be available for participating
0700 *               and non-participating (sync) type requests via the
0710 *               the record buffer extraction (STPRBE) routine, only.
0720 *
0730 *               Determination of the record buffer being for access or
0740 *               update is specified by the caller via the additions 3
0750 *               field (see above).
0760 *
0770 ************************************************************************
0780 DEFINE DATA PARAMETER
0790  01  REQ-TYPE   (A1)
0800  01  P-PROC     (A8)              /* Procedure name
0810  01  P-OPTIONS  (A8)
0820  01  REDEFINE P-OPTIONS
0830    02  P-TYPE   (A1)             /* Async versus sync procedure
0840    02  P-PARMS  (A1)             /* Parm type for procedure
0850    02  P-RECB   (A1)             /* Rec buffer access
0860  01  P-LEN1     (P3)             /* Length of Parm1
0870  01  P-PARM1(A1/1:V)             /* Variable-length parameter 1
0880  01  P-LEN2     (P3)             /* Length of Parm2
0890  01  P-PARM2(A1/1:V)             /* Variable-length parameter 2
0900  01  P-LEN3     (P3)             /* Length of Parm3
0910  01  P-PARM3(A1/1:V)             /* Variable-length parameter 3
0920  01  P-LEN4     (P3)             /* Length of Parm4
0930  01  P-PARM4(A1/1:V)             /* Variable-length parameter 4
0940  01  P-LEN5     (P3)             /* Length of Parm5
0950  01  P-PARM5(A1/1:V)             /* Variable-length parameter 5
0960  01  P-MSG     (A72)             /* Message corresponging to the RESP
0970  01  RESP       (N4)             /* Response code of proc request
0980            LOCAL USING STPLCB
0990            LOCAL
1000  01  SUB        (I2)
1010  01  SUB1       (I2)
1020  01  SUB2       (I2)
1030  01  SUB3       (I2)
1040  01  FB        (A64)
1050  01  REDEFINE FB
1060    02  FB-FIELD  (8)
1070      03  FB-FLD (A3)
1080      03  FB-LEN (N3)
1090      03  FB-COMM(A1)
1100  01  RB    (A1/1000)             /* Max length for all parms
1110  01  W-ADD3     (A8)
1120  01  REDEFINE W-ADD3
1130    02  W-TYPE   (A1)
1140    02  W-PARMS  (A1)
1150    02  W-RECB   (A1)
1160  01  #LENGTH    (B2)
1170  01  REDEFINE #LENGTH
```

```
1180    02  #LENG  (A1/2)
1190  01  W-LENG   (P3/5)
1200 END-DEFINE
1210 FORMAT PS=0
1220 *
1230 MOVE P-LEN1 TO W-LENG(1)
1240 MOVE P-LEN2 TO W-LENG(2)
1250 MOVE P-LEN3 TO W-LENG(3)
1260 MOVE P-LEN4 TO W-LENG(4)
1270 MOVE P-LEN5 TO W-LENG(5)
1280 *
1290 * In this example, we will say that each parameter has an individual
1300 * maximum length of 200; however, the limit may be established as a
1310 * total of all parameters. Since our max. record buffer is 1000, the
1320 * maximum of all parameters cannot exceed 1000. This may be changed as
1330 * required by the user.
1340 *
1350 FOR SUB1 1 5                       /* Validate all parameter lengths
1360  IF W-LENG(SUB1) > 16448          /* Does length contain X'4040'
1370    RESET W-LENG(SUB1)             /*   yes, then must be dummy parm
1380  IF W-LENG(SUB1) > 200            /* For our example we limit the length
1390    DO
1400      MOVE 15   TO RESP
1410      MOVE SUB1 TO FB-LEN(SUB1)
1420      COMPRESS 'Invalid Length for Parameter' FB-LEN(SUB1)
1430               '. Max is 200.' INTO P-MSG
1440      ESCAPE ROUTINE               /* Terminate processing with error
1450    DOEND
1460 CLOSE LOOP
1470 *
1480 * Now we validate the parameters, as required. Of course, these may
1490 * be changed as per the user's requirement and may vary from one stored
1500 * procedure link routine to another.
1510 *
1520 IF P-PROC = ' '                   /* Did we get a procedure name?
1530  DO
1540    MOVE 1 TO RESP
1550    MOVE 'Invalid Procedure Name specified' TO P-MSG
1560    ESCAPE ROUTINE
1570  DOEND
1580 IF NOT (P-TYPE = 'A' OR= 'N' OR= 'P' OR= ' ')
1590  DO                              /* Async, Participating, Non-Partic
1600    MOVE 2 TO RESP
1610    MOVE 'Proc Type must be A, N, P or " "' TO P-MSG
1620    ESCAPE ROUTINE
1630  DOEND
1640 IF NOT (P-PARMS = 'C' OR= 'E' OR= 'N' OR= ' ')
1650  DO                              /* Cntrl, Error/Resp, None
1660    MOVE 3 TO RESP
1670    MOVE 'Parameter Type must be C, E, N or " "' TO P-MSG
1680    ESCAPE ROUTINE
1690  DOEND
1700 IF NOT (P-RECB = 'A' OR= 'N' OR= 'U' OR= ' ')
1710  DO                              /* Access, None, Update
1720    MOVE 3 TO RESP
1730    MOVE 'Parameter access must be Access, None or Update' TO P-MSG
1740    ESCAPE ROUTINE
1750  DOEND
1760 IF P-LEN1 < 3                    /* Min. length with inclusive length
1770  DO                              /* Anything less indicates no parm
1780    MOVE 4 TO RESP
1790    MOVE 'First Parameter MUST be valid. Length must be
```

```
3-200' TO P-MSG
1800    ESCAPE ROUTINE
1810  DOEND
1820 *
1830 * Next we merge all the passed parameters into a single contiguous
1840 * buffer which will be used as the record buffer for the call. The
1850 * format buffer will also be set up to indicate the 'structure' of the
1860 * record buffer for use by the invoked procedure.
1870 *
1880 MOVE 1 TO SUB
1890 RESET SUB2
1900 *
1910 FOR SUB1 1 5                        /* Step through all parameters
1920  IF W-LENG(SUB1) < 3               /* Check min. length of a parameter
1930    DO
1940      MOVE '.' TO FB-FLD(SUB1)
1950      ESCAPE BOTTOM                 /* None, so assume we have all parms
1960    DOEND
1970  ADD W-LENG(SUB1)  TO SUB2         /* Get end position
1980  MOVE W-LENG(SUB1) TO SUB3         /* Set index for MOVE statement
1990  DECIDE ON FIRST VALUE OF SUB1     /* Move next parm into the RB
2000     VALUE 1   MOVE P-PARM1 (1:SUB3) TO RB(SUB:SUB2)
2010               MOVE 'P1,' TO FB-FLD(1)
2020     VALUE 2   MOVE P-PARM2 (1:SUB3) TO RB(SUB:SUB2)
2030               MOVE ','   TO FB-COMM(SUB1 - 1)
2040               MOVE 'P2,' TO FB-FLD(2)
2050     VALUE 3   MOVE P-PARM3 (1:SUB3) TO RB(SUB:SUB2)
2060               MOVE ','   TO FB-COMM(SUB1 - 1)
2070               MOVE 'P3,' TO FB-FLD(3)
2080     VALUE 4   MOVE P-PARM4 (1:SUB3) TO RB(SUB:SUB2)
2090               MOVE ','   TO FB-COMM(SUB1 - 1)
2100               MOVE 'P4,' TO FB-FLD(4)
2110     VALUE 5   MOVE P-PARM5 (1:SUB3) TO RB(SUB:SUB2)
2120               MOVE ','   TO FB-COMM(SUB1 - 1)
2130               MOVE 'P5,' TO FB-FLD(5)
2140               MOVE '.'   TO FB-COMM(5)
2150     ANY       ADD W-LENG(SUB1) TO SUB  /* Get new position
2160               MOVE W-LENG(SUB1) TO FB-LEN(SUB1)
2170     NONE      IGNORE
2180  END-DECIDE
2190 *
2200 CLOSE LOOP
2210 *
2220 * Now we set up the CB for the actual stored procedure call.
2230 *
2240 RESET CB                           /* Clear the ACB
2250 MOVE 'STP'   TO CB-CID             /* Command ID
2260 MOVE 'PC'    TO CB-CMD             /* Command Code
2270 MOVE 77      TO CB-DBID            /* Database ID
2280 MOVE 22      TO CB-FNR             /* File Number
2290 MOVE 64      TO CB-FBL             /* FB Length
2300 MOVE 1000    TO CB-RBL             /* RB length
2310 MOVE P-PROC  TO CB-ADD1            /* Stored procedure name
2320 *
2330 * If any options were not passed, we use a pre-specified default.
2340 *
2350 MOVE 'A' TO W-TYPE                  /* Set the default options
2360 MOVE 'C' TO W-PARMS
2370 MOVE 'N' TO W-RECB
2380 IF NOT (P-TYPE = ' ')              /* Should we default to Async?
2390  MOVE P-TYPE TO W-TYPE
2400 IF NOT (P-PARMS = ' ')             /* Should we default to Contrl?
```

```
2410  MOVE P-PARMS TO W-PARMS
2420 IF NOT (P-RECB = ' ')             /* Should we default to None?
2430  MOVE P-RECB TO W-RECB
2440 MOVE W-ADD3  TO CB-ADD3           /* Options for request 2450 *
2460 CALL 'CMADA' USING CB FB RB(1)   /* Invoke the stored procedure
2470 *
2480 IF CB-RSP NE 0
2490  DO
2500    PRINT (CD=YE) 'Resp ..' (YEI) CB-RSP(EM=HH) 'Add2' CB-ADD2(EM=H(4))
2510                  'Add3' CB-ADD3(EM=H(8)) 'Add4' CB-ADD4(EM=H(8))
2520    ESCAPE ROUTINE
2530  DOEND
2540 *
2550 * Now we need to restore the parameters back into the user's area,
2560 * in case the data was modified. This can happen only if the record
2570 * buffer was modifiable; i.e., P-RECB was set to 'U'.
2580 *
2590 IF  CB-RSP = 0                    /* Was everything okay
2600 AND P-RECB = 'U'                  /* Update: Parms may have been updated
2610  DO
2620    MOVE 1 TO SUB
2630    RESET SUB2
2640    FOR SUB1 1 5
2650      ADD  W-LENG(SUB1) TO SUB2
2660      MOVE W-LENG(SUB1) TO SUB3
2670      DECIDE ON FIRST VALUE OF SUB1 /* Restore parm from RB
2680        VALUE 1   ASSIGN P-PARM1 (1:SUB3) = RB(SUB:SUB2)
2690        VALUE 2   ASSIGN P-PARM2 (1:SUB3) = RB(SUB:SUB2)
2700        VALUE 3   ASSIGN P-PARM3 (1:SUB3) = RB(SUB:SUB2)
2710        VALUE 4   ASSIGN P-PARM4 (1:SUB3) = RB(SUB:SUB2)
2720        VALUE 5   ASSIGN P-PARM5 (1:SUB3) = RB(SUB:SUB2)
2730        ANY       ADD W-LENG(SUB1) TO SUB /* Get next position
2740        NONE      IGNORE
2750      END-DECIDE
2760    CLOSE LOOP(2640)
2770  DOEND
2780 *
2790 END
```