ACR/Detail®

User Program Guide

For z/OS®, UNIX®, Linux®, Windows® and IBM® i
Release 4.2

Home Office
1240 East Diehl Road, Suite 400
Naperville, IL 60563
Tel: +1.630.505.1800

www.infogix.com

Publication Number 1610

Publication Date 7/13/10

Contents

Chapter 7 EBCDIC Variable Block Input Source Reader Program (UUPVBIO)

1

Introduction

ACR/Detail can directly access the following types of files: physical sequential, direct spool, VSAM KSDS, and DB2 tables. To enable extraction and reconciliation of data from other file organization types, you can use a user-written file extraction program to perform I/O at ACR/Detail's request.

The remaining chapters in this manual describe the user programs that Unitech provides for file extraction. You can also create your own user programs based on the information provided in this chapter.

This chapter contains the following sections:

Sample Source Code

Source code for a sample user program for file extraction is shown beginning on the following page. This sample program is called UDSUPRG, but any valid program name can be used.

After the sample code, the requirements for such programs are provided.

Sample Source Code

User Program File Extraction Exit Page 1 of 2

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  UDSUPRG.
*****************************************************************
*                                                               *
* THIS IS A SAMPLE PROGRAM FOR IMPLEMENTING USER PROGRAM I/O.   *
*                                                               *
*****************************************************************
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT SAMPLE-FILE
         FILE STATUS IS WS-FILE-STATUS
         ASSIGN TO UT-S-SAMPLE.
 DATA DIVISION.
 FILE SECTION.
 FD  SAMPLE-FILE
     BLOCK CONTAINS 0 RECORDS
     DATA RECORD IS SAMPLE-RECORD.
 01  SAMPLE-RECORD           PIC X(80).
 WORKING-STORAGE SECTION.
 01  FILLER.
     05  FILLER                          PIC X(040)
     VALUE   'WORKING-STORAGE FOR UDSUPRG STARTS HERE.'.
     05  WS-FILE-STATUS                  PIC 9(002).
     05  FILLER REDEFINES WS-FILE-STATUS PIC X(002).
         88  WS-FILE-STATUS-OK           VALUE '00'.
         88  WS-FILE-STATUS-EOF          VALUE '10'.
 LINKAGE SECTION.
 COPY UPRGLUNI.

*** A MAXIMUM OF 32767 BYTES MAY BE RETURNED TO ACR/ACRDETAIL ***
 01  UPRG-RECORD-AREA.
     05  FILLER                  PIC X(32767).

 EJECT
 PROCEDURE DIVISION  USING UPRG-USING UPRG-RECORD-AREA.
     MOVE ZERO TO UPRG-RC.
     IF UPRG-REQ-GETNEXT
         PERFORM 2000-GETNEXT
     ELSE IF UPRG-REQ-OPEN
        PERFORM 1000-OPEN
     ELSE IF UPRG-REQ-CLOSE
        PERFORM 3000-CLOSE.

     GOBACK.

 1000-OPEN.
     OPEN INPUT SAMPLE-FILE.

**** ANY STATUS OTHER THAN 00 RESULTS IN ERROR 1000 BEING
**** RETURNED TO ACR/UDSAIL.

     IF WS-FILE-STATUS-OK
         NEXT SENTENCE
     ELSE
         MOVE +1000 TO UPRG-RC.
```

User Program File Extraction Exit Page 2 of 2

```
 2000-GETNEXT.
     READ SAMPLE-FILE.

**** ANY STATUS OTHER THAN 00 (OK) OR 10 (EOF) RESULTS IN
**** ERROR 2000 BEING RETURNED TO ACR/UDSAIL.

     IF WS-FILE-STATUS-OK
         MOVE SAMPLE-RECORD TO UPRG-RECORD-AREA
     ELSE IF WS-FILE-STATUS-EOF
         MOVE +1 TO UPRG-RC
     ELSE
         MOVE +2000 TO UPRG-RC.

 3000-CLOSE.
     CLOSE SAMPLE-FILE.
```

```
****************************************************************
* USER PROGRAM FILE EXTRACTION LINKAGE AREA                    *
****************************************************************
*
 01  UPRG-USING.
     05  UPRG-MODULE             PIC X(008).
     05  UPRG-RLSE               PIC X(004).
         88  UPRG-RLSE-OK        VALUE '0400'.
     05  UPRG-REQ                PIC S9(04) COMP.
         88  UPRG-REQ-OPEN       VALUE +0001.
         88  UPRG-REQ-GETNEXT    VALUE +0002.
         88  UPRG-REQ-CLOSE      VALUE +0003.
     05  UPRG-RC                 PIC S9(04) COMP.
         88  UPRG-RC-OK          VALUE +0000.
         88  UPRG-RC-EOF         VALUE +0001.
         88  UPRG-RC-ERR         VALUES +0002 THRU +9999.
     05  UPRG-FILE-ID.
         10  UPRG-FILE-DDNAME    PIC X(008).
         10  UPRG-FILE-QUAL      PIC X(002).
     05  UPRG-DATA               PIC X(018).
     05  FILLER                  PIC X(160).
     05  UPRG-END                PIC X(008).
```

Requirements for User Programs for File Extraction

User programs for file extraction provide file or database I/O for ACR/Detail to use in processing file definitions. ACR/Detail treats these programs like a sequential file. The user program for a particular set of file definitions will be called as follows:

1. A single “open file” call (UPRG - REQ = +1). ACR/Detail passes the file definition's file ID (UPRG- FILE-ID) and any user program data specified on the basic file information record for that file ID. ACR/Detail expects the user program to set the user program return code, indicating success or failure (UPRG - RC = 0 means success; any other value means failure). If the user program indicates failure, no further calls will be made for that File ID.
2. Repetitive “get next record” calls (UPRG - REQ = +2). ACR/Detail will continue to make these calls as long as the user program returns a successful user program return code (UPRG - RC = +0). For each successful call, ACR/Detail expects the user program to set UPRG-RECORD-AREA to the data it should use for extraction.
   When the user program has no more records to return, it should set the user program return code to “end-of-file” (UPRG - RC = +1).
   If the user program encounters an error for which ACR/Detail should abort processing for the file ID, it can set the user program return code to any value greater than +1 (UPRG - RC = +2 through +9999).
3. A single “close file” call (UPRG - REQ = +3). ACR/Detail ignores the user program return code from this call. The user program is expected to close the file and/or perform the appropriate termination processing.

Note: The close call will be made even if the last “get next record” returned an error condition. In other words, if the user program indicated success for the “open file” call, it will get a “close file” call.

2

Delimited Field Access Program (UUPDLIM)

The Delimited Field Access User Program (UUPDLIM) can be used with ACR/Detail to extract data from files containing records of variable length fields that are separated by one or more delimiters. A common example of this type of file is a Comma Separated Value (CSV) file popular on the PC.

UUPDLIM can read files of up to 32,767 bytes.

This chapter consists of the following sections:

Instructions

To use UUPDLIM, you need to do the following:

1. Define a file ID for the variable length record file just as you would for any other ACR/Detail input file. This file ID will be the DDname you will specify in the execution JCL.
2. Define the following parameter for Basic File Information:

   File Organization: `UP`
3. Identify the following user program name for User Program Information:

   User Program Name: `UUPDLIM`
4. (Required) Specify an output parameter for the User Program Information.

   User Program Data: `/S /D<delimiter info> /E`

   /S—Sequence ID. An optional flag that prefixes the output record with a sequence ID consisting of an 8-digit physical record number plus the 8-digit field number.

   /D—Delimiter. A required parameter that indicates the delimiter(s) separating fields in the records. The <delimiter info> has the following structure:

```
[enclosing character][delimiter character(s)][enclosing character]
```

The enclosing character can be any character, excluding blank and delimiter characters.

For a file with fields separated by an equal sign (=) or a comma (,), the <delimiter info> could be specified as either of the following:

`/D'=,'`

/D$=,$

For a file with fields separated by a blank or a single quote, the <delimiter info> could be specified as follows:

`/D- '-`

/E—Enclosure character. An optional parameter indicating that values between 2 of these characters are all one field and embedded delimiters should be ignored.

If you do not specify any delimiter information, UUPDLIM will return the entire physical record, as it would if File Organization of `PS` had been specified.

UUPDLIM Examples

Sample File

The following example shows the record layout for a sample file.

```
         1         2         3         4
12345678901234567890123456789012345678901
TYPE=HEADER,1,02,HELLO,03
TYPE=DETAIL,1,2,33,4,55,6
TYPE=DETAIL,123,456,789,012,345,678
```

Example of Output Using /D"=,"

The following example shows what would be returned to ACR/Detail if the sample file shown above were used and the output parameter in UUPDLIM was set as follows:

Parameter Setting

```
USER PROGRAM: UUPDLIM  DATA: /D"=,"
```

Output

```
         1         2         3         4
12345678901234567890123456789012345678901
TYPE
HEADER
1
02
HELLO
03
TYPE
DETAIL
1
2
33
4
55
6
TYPE
DETAIL
123
456
789
012
345
678
```

Output using /S /D"=,"

The following example shows what would be returned if the sample file was used with the following output parameter setting:

Parameter Setting

```
USER PROGRAM: UUPDLIM  DATA: /S /D"=,"
```

Output

```
         1         2         3         4
12345678901234567890123456789012345678901
00000001000000001TYPE
00000001000000002HEADER
00000001000000031
00000001000000402
00000001000000005HELLO
00000001000000603
00000002000000001TYPE
00000002000000002DETAIL
00000002000000031
00000002000000042
00000002000000533
00000002000000064
00000002000000755
00000002000000086
00000003000000001TYPE
00000003000000002DETAIL
00000003000000003123
00000003000000004456
00000003000000005789
00000003000000006012
00000003000000007345
00000003000000008678
```

3

XML File Access Program (UUPXML)

The XML File Access User Program (UUPXML) can be used with ACR/Detail to extract data from XML files and pass it to ACR/Detail extraction. The program formats the XML so that it can be read by ACR/Detail extraction, similar to how UUPDLIM processes delimited files.

This chapter consists of the following sections:

About UUPXML

UUPXML can read files of up to 32,767 bytes.

For each record (line) in the XML file that is parsed, individual "records" will be passed to ACR/Detail's extraction process. Each "record" will be a tag name, attribute name, attribute value, or tag value, and each is passed on a separate line.

End tags and the initial "?XML version" tag will be not be passed to ACR/Detail.

For example, suppose your XML line looks like this:

```
<myTag myAttr="myAValue">myRealValue</myTag>
```

(For now, we will assume that no parameters have been specified. The /S and /X parameters will be explained below.)

The "records" for this sample line would be passed to ACR/Detail in the same sequence as they appear in the input line, as follows:

```
TNmyTag             (tag name)
ANmyAttr            (attribute name)
AVmyAValue          (attribute value)
TVmyRealValue       (tag value)
```

Instructions

To use UUPXML, do the following:

1. Define a file ID for the XML file just as you would for any other ACR/Detail input file. This file ID will be the DDname you will specify in the execution JCL.
2. Define the following parameter for Basic File Information:
   File Organization: **UP**
3. Identify the following user program name in the user program information:
   User Program Name: **UUPXML**
4. Optionally specify the /S parameter, the /X parameter, or both in the program data parameter field in the user program information. The following entries are valid:
   - Program Data Parameter: **/S**
   - Program Data Parameter: **/X**
   - Program Data Parameter: **/S /X**

"Records" Passed Using /S Parameter Alone

If you use the /S parameter alone, three sequence numbers will be passed to the file, as follows:

- Columns 1-8 will show the number of root tags in the file, which is also the number of XML documents found. If the file contains only one XML document, this number will be 00000001. If the file contains multiple XML documents or if XML files are concatenated, this number will be 00000001 for the first document, 00000002 for the second, and so on.
- Columns 9-16 will show a count of the indents you would see if the file were viewed in a browser. This is the depth of the tag. Each tag name will increment this number. Each end tag will decrement this number.
- Columns 17-24 will show a record sequence. This number will be incremented for each item processed at this tag depth.

Sample "Records" Using /S Parameter Alone

If your XML line is the same as in the previous example:

```
<myTag myAttr="myAValue">myRealValue</myTag>
```

...the 4 records would look like this:

```
00000001000000010000001TNmyTag         (tag name record)
00000001000000010000002ANmyAttr        (attribute name record)
00000001000000010000003AVmyAValue      (attribute value record)
00000001000000010000004TVmyRealValue   (tag value record)
```

"Records" Passed Using Both /S and /X Parameters

If you use both the /S and /X parameters, in addition to being passed to ACR/Detail extraction, each parsed item is displayed in the ACR/Detail sysout as a "record." The record will include sequence numbers, the type of data, and the data itself.

If you use both parameters, it will be easier to write the rules because you can refer to the display or to a printout of the display.

Note: For the IBM i, the parsed item is included in the job log, which is created as a spool file in an OUTQ and can be printed at the end of the job.

Sample "Records" Using both /S and /X parameters

Suppose your XML file looks like this:

```
  <?xml version="1.0" encoding="UTF-8" ?>
- <infogixdata>
  - <rpt type="FREEFORM" version="4.0"
     - <ji>
           <prod>ACRD</prod>
           <job>BASIC</job>
           <step>JOB</step>
           <qual >  </qual>
           <pid />
           <sdte>20071113</sdte>
           <sdme>114822</sdme>
           <cyc>00000008</cyc>
           <run />
           <tz />
           <fmt22>N</fmt22>
       </ji>
       <record>this is the whole report, just a test</record>
  - </rpt>
</infogixdata>
```

The displayed "records" will look like this:

```
00000001000000010000001TNinfogixdata
00000001000000020000001TNrpt
00000001000000020000002ANtype
00000001000000020000003AVFREEFORM
00000001000000020000004ANversion
00000001000000020000005AV4.0
00000001000000020000006TV
00000001000000030000001TNji
00000001000000040000001TNprod
00000001000000040000002TVACRD
00000001000000040000003TNjob
00000001000000040000004TVBASIC
00000001000000040000005TNstep
00000001000000040000006TVJOB
00000001000000040000007TNqual
00000001000000040000008TV
00000001000000040000009TNpid
00000001000000040000010TNsdte
00000001000000040000011TV20071113
00000001000000040000012TNsdme
00000001000000040000013TV114822
00000001000000040000014TNcyc
00000001000000040000015TV00000008
00000001000000040000016TNrun
00000001000000040000017TNtz
00000001000000040000018TNfmt22
00000001000000040000019TVN
00000001000000030000001TNrecord
00000001000000030000002TVThis is the whole report, just a test
```

4

EDI User Program (UUPEDIF)

The EDI User Program (UUPEDIF) can be used with ACR/Detail to extract data from electronic data interchange (EDI) files in X12 format.

This chapter consists of the following sections:

Instructions

To use UUPEDIF, you need to do the following:

1. Define a file ID for the variable length record file just as you would for any other ACR/Detail input file. This file ID will be the DDname you will specify in the execution JCL.
2. Define the following parameter for Basic File Information:
   File Organization: UP
3. Identify the user program name for User Program Information:
   User Program Name: UUPEDIF
4. (Optional) Specify an output parameter for the User Program Information:
   User Program Data: /B /G or /S
   /B—Places the state of the current batch in column 1 of the output record. Additional characters in the record are moved one character to the right.
   A batch is a sequence of data beginning with an ISA group and ending with an IEA group. A batch can have the following states:
   B—At the beginning of a batch
   I—In a batch
   E—At the end of a batch
   /G—Places the state of the current EDI group in front of the output record.

A group is the sequence of fields beginning with a 3-character identifier, and ending with the end-of-group delimiter. The state will be shown in column 1 if /G is used without /B. The state will be shown in column 2 if used with /B, regardless of the order in which the parameters are specified. A group can have the following states:

B—At the beginning of a group

I—In a group

E—At the end of a group

/N—Processes an EDI file where a new group starts on a new line.

/S—Places a group sequence ID at the beginning of each output record, but after any “state” flags. The group sequence ID is 8 characters long, consisting of the 3-character group ID and the 5-digit sequence number of the delimited field within that sequence ID.

UUPEDIF can identify ISA segment records that are irregular in either of the following ways:

- They do not begin on card column 1.
- They are not preceded by an end of segment delimiter (ESD).

Examples

Sample File

UUPEDIF returns a separate "record" for each delimited field in the EDI file. The following example shows an abbreviated, native EDI file.

```
ISA*00*          *00*          *01*INFOGIX        *01*INFOGIX       *930922*195
0*U*00200*000000091*0*T*>|GS*IN*INFOGIX     *INFOGIX
*930922*1950*91*X*003010|S
T*810*2085|IEA*1*000000091|
```

Example of Results with No Output Parameters Set

The following example shows what would be returned to ACR/Detail by the sample input file if UUPEDIF were run with no output parameters set:

```
         1         2         3         4
1234567890123456789012345678901234567890
----------------------------------------
ISA
00

00

01
INFOGIX
01
INFOGIXT
930922
1950
U
00200
000000091
0
T
>
GS
IN
INFOGIX
INFOGIX
930922
1950
91
X
003010
ST
810
2085
IEA
1
000000091
```

Note: An element that has no data, but is contained within delimiters, will be shown as blank, but will be accounted for.

Example of Results with the /S Output Parameter

The following example shows what would be returned to ACR/Detail by the sample input file if UUPEDIF were run with the /S output parameter:

```
         1         2         3         4
1234567890123456789012345678901234567890
----------------------------------------
ISA00000ISA
ISA0000100
ISA00002
ISA0000300
ISA00004
ISA0000501
ISA00006INFOGIX
ISA0000701
ISA00008INFOGIXT
ISA00009930922
ISA000101950
ISA00011U
ISA0001200200
ISA000013000000091
ISA000140
ISA00015T
ISA00016>
GS 00000GS
GS 00001IN
GS 00002INFOGIX
GS 0000INFOGIX
GS 00004930922
GS 000051950
GS 0000691
GS 00007X
GS 00008003010
ST 00000ST
ST 00001810
ST 000022085
IEA00000IEA
IEA000011
IEA00002000000091
```

Example of Results with the /B Output Parameter

The following example shows what would be returned to ACR/Detail by the sample input file if UUPEDIF were run with the `/B` output parameter:

```
         1         2         3         4
1234567890123456789012345678901234567890
----------------------------------------
BISA
I00
I
I00
I
I01
IINFOGIX
I01
IINFOGIXT
I930922
I1950
IU
I00200
I000000091
I0
IT
I>
IGS
IIN
IINFOGIX
IINFOGIX
I930922
I1950
I91
IX
I003010
IST
I810
I2085
IIEA
I1
E000000091
```

Example of Results with the /G Output Parameter

The following example shows what would be returned to ACR/Detail by the sample input file if UUPEDIF were run with the `/G` output parameter:

```
         1         2         3         4
12345678901234567890123456789012345678901
----------------------------------------
BISA
I00
I
I00
I
I01
IINFOGIX
I01
IINFOGIXT
I930922
I1950
IU
I00200
I000000091
I0
IT
E>
BGS
IIN
IINFOGIX
IINFOGIX
I930922
I1950
I91
IX
E003010
BST
I810
E2085
BIEA
I1
E000000091
```

Example of Results with the /B, /G, and /S Output Parameters

The following example shows what would be returned to ACR/Detail by the sample input file if UUPEDIF were run with the `/B, /G, and /S` output parameters:

```
         1         2         3         4
1234567890123456789012345678901234567890
----------------------------------------
BBISA00000ISA
IIISA0000100
IIISA00002
IIISA0000300
IIISA00004
IIISA0000501
IIISA00006INFOGIX
IIISA0000701
IIISA00008INFOGIXT
IIISA00009930922
IIISA000101950
IIISA00011U
IIISA0001200200
IIISA00013000000091
IIISA000140
IIISA00015T
IEISA00016>
IBGS 00000GS
IIGS 00001IN
IIGS 00002INFOGIX
IIGS 00003INFOGIX
IIGS 00004930922
IIGS 000051950
IIGS 0000691
IIGS 00007X
IEGS 00008003010
IBST 00000ST
IIST 00001810
IEST 000022085
IBIEA00000IEA
IIIEA000011
EEIEA00002000000091
```

Examples

5

Variable Length Record User Program (UUPVREC)

The Variable Length Record User Program (UUPVREC) can be used with ACR/Detail to extract data that occurs multiple times in a variable length record. The records can consist of a fixed header area and a variable number of line areas. For example, a health insurance claim form consists of header information, such as names and addresses, and multiple occurrences of services performed.

For each occurrence of line data in the physical record, UUPVREC returns to ACR/Detail a record consisting of the header portion of the record plus a single occurrence of the line data.

The advantages of this program include the following:

- UUPVREC allows ACR/Detail to read each occurrence of "line" information as a separate logical record, making it much easier to apply controls.
- Variable length records that exceed the 10,000 character maximum that ACR/Detail extraction can directly access become accessible.

This chapter consists of the following sections:

Instructions

To use UUPVREC with ACR/Detail, you need to do the following:

1. Determine the characteristics of the input file.
   The following information is necessary for using UUPVREC:
   - Starting position of the first occurrence of the variable line area.
   - Length of each occurrence of the variable line area.
   - The position, length, and field format of the field within the fixed header that indicates how many occurrences of variable data there are for a given record. All positions and lengths should be based on the record layout without consideration for the Record Descriptor Word (length) data.

     For example, if positions 1-4 are length information, 5-104 are the header information, and 105-xxxx are the variable lines, then you would indicate to UUPVREC that the variable lines start in position 101.
2. Define a File ID for the variable length record file just as you would for any other ACR/Detail input file. This File ID will be the DDname you will specify in the execution JCL.
3. Define the following parameter for Basic File Information:
   File Organization: `UP`
4. Identify the following user program name for User Program Information:
   User Program Name: `UUPVREC`
5. Specify User Program Data for the User Program Information. See "Record Layout for User Program Data" on page 27
6. If you receive #UDX033E: USER PROGRAM OPEN ERROR, see "Troubleshooting" on page 29.

Record Layout for User Program Data

Record Layout Table

The layout for the user program data is as follows:

Pos	Length	Format	Description	Values
1-4	4	Numeric	Starting position of first variable occurrence	
5-8	4	Numeric	Length of one variable occurrence	
9-12	4	Numeric	Starting position of Number of Occurrences field	
13-14	2	Numeric	Length of Number of Occurrences field	
15	1	Numeric	Format of Number of Occurrences field	1,3,4, or 5[1]
16	1	Character	Place physical record number in front of data[2]	Y or blank
17	1	Character	Place varying occurrence number in front of record data[2]	Y or blank
The footnote is explained in the following section.				

Notes to the Record Layout

[1]Values for Format Number of Occurrences (position 15) are as follows:

1	Display Numeric	(PIC9(nn))
3	Packed	(PIC S9(nn) COMP-3)
4	Signed Numeric	(PIC S9(nn)
5	Binary	(PIC S9(nn) COMP)

[2]Values for the Place Physical Record Number in Front of Data field (position 16) and Place Varying Occurrence Number in Front of Record Date (Position 17) are as follows:

If both the Physical Record Number and Occurrence Number output options are blank, the records are returned with the header area beginning in position 1, followed immediately by the varying occurrence area.

If only the Physical Record Number option is Y, the physical record number is output in the first eight positions (Display Numeric) of the record area, followed immediately by the actual record information (starting in position 9).

If only the Number of Occurrences option is Y, the occurrence number is output in the first eight positions (Display Numeric) of the record area, followed immediately by the actual record information (starting in position 9).

If both the Physical Record Number and the Occurrence Number options are Y, the physical record number is placed in the first eight positions of the record area; the occurrence number is placed in positions 9-16; the actual record information follows immediately, beginning in position 17.

Troubleshooting

When UUPVREC processes the open request from ACR/Detail, it applies edits again the user program data. If errors are detected, a return code value is set to notify ACR/Detail. ACR/Detail reports the return code via the following message:

#UDX033E: USER PROGRAM OPEN ERROR (RC=*nnnn*) (PROGRAM=UUPVREC)

where *nnnn* is one of the following:

Return Code	Reason
1001	The starting position of the first variable occurrence was not numeric (positions 1-4).
1002	The length of one variable occurrence was not numeric (positions 5-8).
1003	The position of the Number of Occurrences field was not numeric (positions 9-12).
1004	The length of the Number of Occurrences field was not numeric (positions 13-14).
1005	The format of the Number of Occurrences field was not numeric (position 15).
1010	The starting position of the first variable occurrence was zero.
1011	The length of one variable occurrence was zero.
1012	The position of the Number of Occurrences field was zero.
1013	The length of the Number of Occurrences field was zero.
1014	The format of the Number of Occurrences field contained an invalid value (only `1`, `3`, `4`, and `5` are valid).
1015	The length of an output "record" from UUPVREC would exceed the maximum of 10,000 supported by ACR/Detail. The results of the following formula cannot exceed 10,000: the starting position of the first variable occurrence + the length of a variable occurrence -1 (+8 if the option of returning physical record number is set) (+8 if the option of returning varying occurrence number is set).

6

EBCDIC Fixed Block Input Source Reader Program (UUPFBIO)

This program can be used with ACR/Detail to make mainframe EBCDIC fixed block input files accessible to ACR/Detail on UNIX, Linux, and Windows.

Instructions

To use the program, specify the following information in the graphical interface for ACR/Detail:

1. From Input Source View, select File Description to open the File Description dialog box.
2. Enter @EBC in the first 4 characters of the **File Description** field.
3. Select File Organization to open the File Organization dialog box and complete the fields as follows:

   File Organization/Type: User Program Accessed File.

   Program Data: The record length in 5-digit format.

   Program Name: UUPFBIO.

Instructions

7

EBCDIC Variable Block Input Source Reader Program (UUPVBIO)

This program can be used with ACR/Detail to make mainframe EBCDIC variable block input files accessible to ACR/Detail on Windows, UNIX, and Linux.

Downloading the File

Download the file in binary format, but select CR/LE to retain the end of line markers.

For example, if you use RUMBA for your mainframe emulator, use the following download settings:

1. On the Copy From PC to Host dialog, for **Data Type**, select No translation.
2. Select CR/LE to retain the carriage returns. (Do not select To ASCII).

Specifying the Program in the Graphical Interface

To use the downloaded program, specify the following information in ACR/Detail Client or ACR/Workbench:

1. From Input Source View, select File Description to open the File Description dialog box.
2. Enter @EBC in the first 4 characters of the **File Description** field.
3. Select File Organization to open the File Organization dialog box and complete the fields as follows:

   File Organization/Type: User Program Accessed File.

   Program Name: UUPVBIO.