Using PSF DSS User-Exit Programs

Back to Administrator procedures

This topic describes the sample PSF DSS user-exit programs supplied with Infoprint Manager and shows how you can create your own PSF DSS user-exit programs. These user-exit programs apply to printers connected to Infoprint Manager through the PSF device support subsystem (DSS).

This section contains the following topics:

- "Supported types of PSF DSS user-exits"
- "Sample PSF DSS user-exit programs" on page 2
- "Creating and Using Your Own PSF DSS User-Exit Programs" on page 3
- "Compiling and Installing the User-Exit Program" on page 3
- "Header Page, Trailer Page, and Separator Page User-Exit Program Inputs and Outputs" on page 4
- "Accounting and Audit User-Exit Program Inputs and Outputs" on page 8
- "Input Data User-Exit Program Inputs and Outputs" on page 10
- "Ouput Data User-Exit Program Inputs and Outputs" on page 12
- "Structure of a User-Exit Program" on page 13
- "User-exit programs for the line data transform" on page 13
- "Attributes of the Line Data Input File" on page 20
- "Using the iconv command to convert code pages" on page 21

Supported types of PSF DSS user-exits

Infoprint Manager supports dynamically loaded user-exit programs. These programs are loaded with Infoprint Manager during initialization and are called at various points during job processing.

Infoprint Manager allows seven different types of PSF DSS user exits. Infoprint Manager calls the exits in the following order:

1. Header (Start) page user exit
2. Separator page user exit
3. Accounting user exit
4. Audit user exit
5. Input data user exit
6. Output data user exit
7. Trailer (End) page user exit

Note: Refer to the Setting up to use MVS Download topic for information about the sample user-exit program provided for MVS Download.

Sample PSF DSS user-exit programs

Infoprint Manager supplies sample PSF DSS user-exit programs for the header, separator, and trailer pages, and for accounting and auditing data. The sample header, separator, and trailer page user-exit programs generate Advanced Function Presentation (AFP) data stream pages. The sample accounting and auditing user-exit programs generate ASCII data in a report file format. Infoprint Manager also supplies sample user-exit programs for input data and output data; however, these user-exit programs do not perform any function. You can use all of these sample programs as they are or you can modify them.

If you choose to modify the existing user-exit programs, the source code for each one is located in the *install_path***exits directory**. The *install_path* is the directory that Infoprint Manager is installed in. If you don't know the install path, you can find it in the **Management Console**. Open the Management Console, click **Edit -> Change Service Configuration** and find the **Install path** field.

Table 1 lists the sample user exit programs supplied with Infoprint Manager.

Table 1. Sample user exit programs

Type of user-exit	File Name	Description
Header Page	ainuxhdr.c	Generates brief style start sheet.
	ainuxhdr2.c	Generates full style start sheet.
	ainuxhdrp.c	Generates start sheet with job ticket information.
	ainuxhdrx.c	Generates start sheet without vertical lines.
	pduxblkh.c	Generates a blank start sheet.
Separator Page	ainuxsep.c	Generates brief style separator sheet
	ainuxsep2.c	Generates full style separator sheet.
	ainuxsepp.c	Generates separator sheet with job ticket information.
	ainuxsepx.c	Generates separator sheet without vertical lines.
	pduxblks.c	Generates a blank separator sheet.
Trailer Page	ainuxtlr.c	Generates brief style end sheet.
	ainuxtlr2.c	Generates full style end sheet.
	ainuxtlrp.c	Generates start sheet with job ticket information.
	ainuxtlrx.c	Generates end sheet without vertical lines.
	pduxblkt.c	Generates a blank end sheet.

Table 1. Sample user exit programs (continued)

Type of user-exit	File Name	Description
Accounting	ainuxacc.c	Generates brief style accounting sheet.
	ainuxacc2.c	Generates full style accounting sheet.
	ainacclog.c	Generates accounting log information.
Auditing	ainuxaud.c	Generates brief style audit sheet.
	ainuxaud2.c	Generates full style audit sheet.
	ainuxaudp.c	Generates audit sheet with job ticket information.
	ainaudlog.c	Generates audit log information.
Accounting log	ainuxaccp.c	Generates accounting sheet log with job ticket information.
Input Data	ainuxind.c	Provides template for writing your own user-exit program.
Output Data	ainuxout.c	Provides template for writing your own user-exit program.

Note: Refer to "User-exit programs for the line data transform" on page 13 for sample user-exit programs used with the line data transform.

Creating and Using Your Own PSF DSS User-Exit Programs

You can create your own PSF DSS user-exit program at any time; however, you must perform additional tasks to produce a PSF DSS user-exit executable program. You must have the Microsoft Visual C++ Program, Version 5.0 or higher, installed on your Infoprint NT server in order to compile and build user-exit programs. Be sure to apply the latest service available.

You can create your own user-exit program in one of the following ways:

- Copy one of the sample user-exit programs, rename it, and modify it using the Microsoft Visual C++ Program.
- Write your own user-exit program. You can include the variable data shown in the following sections, for example, `USERID`; but Infoprint cannot process variable data that is not shown in this topic.

Compiling and Installing the User-Exit Program

You can only specify one user-exit program for each function name provided by Infoprint per actual destination.

The function name of the user-exit must be one of the following (in uppercase), because that is the entry point in the module:

- ACCOUNTING
- AUDIT

- HEADER
- SEPARATOR
- TRAILER
- INDATA
- OUTDATA

Activating the User-Exit Program

Before you can use the new user-exit program, you must activate the program.

To perform this task from the **Infoprint Manager Administration GUI**, you must select the printer that you want to associate with an auxiliary sheet (in this case, `prt1`) and then use the **Printer —> Properties** menu to access the **Printer Properties** notebook. To complete this task, refer to the online help topic **Using auxiliary-sheet objects** in the **Infoprint Manager Administration GUI**.

To perform this task from the command line, use the click the Windows **Start** button and select **Programs—>Command Prompt** path and perform the following procedure.

1. To create the auxiliary-sheet object, type:

   ```
   pdcreate -c aix-sh -x psf-exit-prog-name=fullpath\exit_name
   server1:auxiliary_sheet_name
   ```
2. Disable the actual destination (in this case, `prt1`).

   ```
   pddisable prt1
   ```
3. Add this new auxiliary-sheet object (in this case, an audit exit) to the actual destination.

   ```
   pdset -cp -x dest -x audit-exit=auxiliary_sheet_name prt1
   ```
4. Enable the actual destination (in this case, `prt1`).

   ```
   pdenable prt1
   ```

Header Page, Trailer Page, and Separator Page User-Exit Program Inputs and Outputs

Because the Infoprint start page and trailer page user-exit programs require the same inputs and produce the same outputs, the program descriptions are similar. The Infoprint separator page user-exit program differs only in having access to the copy counter output information.

There is one invocation of the separator user-exit prior to every copy of the job data. This assumes that the copies were requested using the result-profile attribute rather that the copy-count attribute (for which the multiple copies are merged into one job file).

The source code (in the C programming language) for the sample header page user-exit programs, trailer page user-exit programs, and separator page user-exit programs are located in the *install_path*`\exits` directory. These sample programs generate an AFP data stream page.

The data structures for the header, trailer, and separator page user-exit programs are included with the source code in the *install_path*`\exits\ainuexit.h` file. The code for these structures is shown in "Structure of a User-Exit Program" on page 13.

The declaration of these exits is:

```
void HEADER (HEADER_EXITDATA *exitdata)
void TRAILER (TRAILER_EXITDATA *exitdata)
void SEPARATOR (SEPARATOR_EXITDATA *exitdata)
```

The **HEADER_EXITDATA**, **TRAILER_EXITDATA**, and **SEPARATOR_EXITDATA** input/output parameters contain all the input and output data needed to communicate between Infoprint and the user-exit programs.

The **HEADER_EXITDATA**, **TRAILER_EXITDATA**, and **SEPARATOR_EXITDATA** structures include these fields.

To build the executable program, you must use the Microsoft Visual C++ Program, Version 5.0 pull-down menu.

Fields that provide input information

UserID
The system user ID of the person who submitted the job.

NodeID
The name of the system (host). The backend program sets the NodeID.

JobName
The name of the job. This is the same as the file name in the job.

Job submitters can also specify the job name with the Infoprint job attribute `job-name=name` or `name=name` when they submit a job. If the job name is not specified for files defined as PCL, PostScript, line data, unformatted ASCII, double-byte character set (DBCS) ASCII, or PDF, files when the job is submitted, Infoprint creates a temporary file for the transform output. Infoprint uses the temporary file name for the job name.

SpoolID
The job ID, which is an integer. Infoprint Manager sets the **SpoolID**.

PrinterName
The name of the Infoprint destination.

hab This field is not used.

Date The date the job is printed in `MM/DD/YY` format.

ExtDate
The date the job is printed in `MM/DD/YYYY` format.

Time The time the job is printed in `HH:MM:SS` format.

Distribution
The distribution information provided when the job is submitted.

Other Input Values

The following fields are used by some of the sample programs to provide additional input information. This information can be specified from:

- migrated jobs
- MVS Download jobs
- fields set in the Infoprint Manager Administration GUI **default-document** notebook on the **PSF Information** tab

The field length limits provided below are based on the currently shipped user-exits. Additional field length is normally provided to the user-exits, but would require modification of the exits to display the additional data.

Account
: Identifies the account information provided when the job is submitted. The **Account** field value can be any null-terminated (X'00') character string. You should limit the string to 20 characters or less. Job submitters can specify this same information through the **account-text** document attribute.

Address1
: Identifies the first line of address information provided when the job is submitted. The **Address1** field value can be any null-terminated (X'00') character string. You should limit the string to 57 characters or less. Job submitters can specify this same information through the **address1-text** document attribute.

Address2
: Identifies the second line of address information provided when the job is submitted. The **Address2** field value can be any null-terminated (X'00') character string. You should limit the string to 57 characters or less. Job submitters can specify this same information through the **address2-text** document attribute.

Address3
: Identifies the third line of address information provided when the job is submitted. The **Address3** field value can be any null-terminated (X'00') character string. You should limit the string to 57 characters or less. Job submitters can specify this same information through the **address3-text** document attribute.

Address4
: Identifies the fourth line of address information provided when the job is submitted. The **Address4** field value can be any null-terminated (X'00') character string. You should limit the string to 57 characters or less. Job submitters can specify this same information through the **address4-text** document attribute.

Building
: Identifies the building information provided when the job is submitted. The **Building** field value can be any null-terminated (X'00') character string. You should limit the string to 24 or less characters. Note that job submitters can specify this same information through the **building-text** document attribute.

Department
: Identifies the department information provided when the job is submitted. The **Department** field value can be any null-terminated (X'00') character string. You should limit the string to 24 characters or less. Job submitters can specify this same information through the **department-text** document attribute.

Name
: Identifies the name information provided when the job is submitted. The **Name** field value can be any null-terminated (X'00') character string. You should limit the string to 24 characters or less. Job submitters can specify this same information through the **name-text** document attribute.

Nodeid
: Identifies the nodeid information provided when the job is submitted. The **Nodeid** field value can be any null-terminated (X'00') character string. You

should limit the string to 10 characters or less. Job submitters can specify this same information through the **node-id-text** document attribute.

Passthru
Identifies any other information provided when the job is submitted that will be passed through the user exit to the backend program. Job submitters can also specify the account information with the Infoprint attribute **printer-pass-through='-pa=class=A...'**

The following **Passthru** flags are extracted by the "full" user-exits. Extracting other values from the **Passthru** variable would require modifications to the supplied user-exit programs.

class Identifies the one-character class attribute.

destination
Identifies the one- to eight-character destination attribute.

forms Idenfiifes the one- to eight-character forms attribute.

segmentid
Identifies the one- to ten-character segmentation identifier.

The **Passthru** value can be any null-terminated (X'00') character string no greater than 1,024 characters in length.

Programmer
Identifies the programmer information provided when the job is submitted. The **Programmer** field value can be any null-terminated (X'00') character string. You should limit the string to 24 characters or less. Job submitters can specify this same information through the **programmer-text** document attribute.

Room Identifies the room information provided when the job is submitted. The **Room** field value can be any null-terminated (X'00') character string. You should limit the string to 24 characters or less. Job submitters can specify this same information through the **room-text** Infoprint document attribute.

Title Identifies the title information provided when the job is submitted. The **Title** field value can be any null-terminated (X'00') character string. You should limit the string to 24 characters or less. Job submitters can specify this same information through the **title-text** document attribute.

Userid
Identifies the userid information provided when the job is submitted. The **Userid** field value can be any null-terminated (X'00') character string. You should limit the string to 10 characters or less. Job submitters can specify this same information through the **user-id-text** document attribute.

Fields that provide output information

Copy Indicates which copy is associated with this call to the separator page user-exit. Initially set to 1, it increments by one each time the exit is called. The sample separator page user-exit program generates one separator page for the first copy and all subsequent copies of the job.

PagePointer
Points to a buffer containing the header page or trailer page data produced by the exit.

PageSize
Indicates the size of the header page or trailer page data returned by the

exit. This field is initially set to 0 before the exit is called. If no data is generated by the exit, this field remains set at 0.

PageType
Indicates the type of header or trailer page data (if any) that the exit generates. The field is initially set to 0 (AFP data stream) before the exit is called.

Valid values are:

0 Advanced Function Presentation (AFP) data stream.

1 ASCII data.

Job Completion
Indicates the status returned by the exit. This field is initially set to 0 before the exit is called.

Accounting and Audit User-Exit Program Inputs and Outputs

The Infoprint accounting and audit user-exit programs require the same inputs and produce the same outputs so the program descriptions are similar.

The source code (in the C programming language) for the sample separator page user-exit programs listed at “Supported types of PSF DSS user-exits” on page 1 are located in the *install_path*`\exits` directory.

These sample programs generate an ASCII data stream page. The sample programs store the data into a file you can access with the following executable reporting utilities:

ainurpt1
Provides accounting data for the actual destinations defined, based upon the destination ID.

ainurpt2
Provides accounting data for the actual destinations defined, based upon the userid.

ainurpt3
Provides detailed accounting data for the destinations defined, based upon the particular userid supplied.

ainurpt4
Provides audit data for the actual destinations defined, based upon the destination ID.

ainurpt5
Provides audit data for the actual destinations defined, based upon the userid.

ainurpt6
Provides detailed audit data for the actual destinations defined, based upon the particular userid supplied.

These executable reporting utilities are located in the *install_path*`\exits` folder. By specifying these executable reporting utilities at the DOS command line, you can display data by either destination ID or userid. For example, to report on print requests submitted by a specific user to a actual destination for which the accounting user exit has been activated, you can specify **ainurpt3** and then provide the userid to the program.

Note: You must activate the audit and accounting exits to generate these reports. For more information on activating these exits, see “Activating the User-Exit Program” on page 4.

The data structures for the accounting and audit user-exit programs are included with the source code in the `*install_path*\exits\ainuexit.h` and `*install_path*\exits\ainurpt.h` files. The code for these structures is shown in “Structure of a User-Exit Program” on page 13.

The declarations of these exits are:

```
void ACCOUNTING (ACCOUNTING_EXITDATA *exitdata)
void AUDIT (AUDIT_EXITDATA *exitdata)
```

The **ACCOUNTING_EXITDATA** and the **AUDIT_EXITDATA** input and output parameters contain all the input and output data needed to communicate between Infoprint and the user-exit programs.

The **ACCOUNTING_EXITDATA** and the **AUDIT_EXITDATA** structures include the following fields.

To build the executable program, you must use the Microsoft Visual C++ Program, Version 5.0 pull-down menu.

Fields that provide input information

UserID
: The system user ID of the person who submitted the job.

NodeID
: The name of the system (host). The backend program sets the NodeID.

JobName
: The name of the job. This is the same as the file name in the job.

: Job submitters can also specify the job name with the Infoprint job attribute **job-name=name** or the alias **name=** when they submit a job. If the job name is not specified for files defined as PCL, PostScript, line data, unformatted ASCII, double-byte character set (DBCS) ASCII, or PDF, files when the job is submitted, Infoprint creates a temporary file for the transform output. Infoprint uses the temporary file name for the job name.

SpoolID
: The job ID, which is an integer. Infoprint sets the **SpoolID**.

PrinterName
: The name of the Infoprint destination.

hab This function is not used.

Pages Printed
: Shows the total number of impressions produced for this job.

Bin One Sheets Printed
: Shows the total number of sheets selected from the primary bin.

Bin Two Sheets Printed
: Shows the total number of sheets selected from the secondary bin.

Number of Fonts used
: Shows the total number of fonts used in this job.

Note: For **Number of Fonts used**, **Number of Overlays used**, and **Number of Segments used**, the counters do not include resources contained in the input file. The counters only include resources used before the accounting or audit user-exit is called. Therefore, the resources required for printing error messages and the trailer page do not appear in the resource totals.

Number of Overlays used
Shows the total number of overlays used in this job.

Number of Segments used
Shows the total number of page segments used in this job.

ExtStart Date
Identifies the four-digit date the job started printing.

Start Date
The two-digit date the job started printing.

Start Time
The time the job started printing.

ExtStop Date
The four-digit date the job finished printing.

Stop Date
The two-digit date the job finished printing.

Stop Time
The time the job finished printing.

Optional fields that can provide other input values for the accounting and auditing user-exit are listed in "Other Input Values" on page 5.

Fields that provide output information

PagePointer
Points to a buffer containing the accounting or audit page data produced by the exit.

PageSize
Indicates the size of the accounting or audit page data returned by the exit. This field is initially set to 0 before the exit is called. If no data is generated by the exit, this field remains set at 0.

PageType
Indicates the type of accounting or audit page data (if any) that the exit generates. The field is initially set to 0 (AFP data stream) before the exit is called.

Valid values are:

0	Advanced Function Presentation (AFP) data stream
1	ASCII data

Input Data User-Exit Program Inputs and Outputs

The Infoprint Manager input data user-exit program is used to monitor the print data stream that is coming into Infoprint Manager. This exit is called at the beginning of the job after the header (start) page.

The source code (in the C programming language) for the sample input data user-exit program listed in "Supported types of PSF DSS user-exits" on page 1 is in the `install_path\exits\ainuxind.c` file. This sample program consists of a return code and performs no function.

The data structures for the output data user-exit program are included with the source code in the `install_path\exits\ainuexit.h` file. The code for these structures is shown in "Structure of a User-Exit Program" on page 13.

The declaration of this exit is:

```
void INDATA (INDATA_EXITDATA *exitdata)
```

The **INDATA_EXITDATA** input/output parameter contains all the input and output data needed to communicate between Infoprint and the user-exit program.

The INDATA_EXITDATA structure includes the following fields.

To build the executable program, you must use the Microsoft Visual C++ Program, Version 5.0 pull-down menu.

Fields that provide input information

UserID
The system user ID of the person who submitted the job.

NodeID
The name of the system (host). The backend program sets the NodeID.

JobName
The name of the job. This is the same as the file name in the job.

Job submitters can also specify the job name with the Infoprint job attribute **job-name=name** when they submit a job. If the job name is not specified for files defined as PCL, PostScript, line data, unformatted ASCII, double-byte character set (DBCS) ASCII, or PDF, files when the job is submitted, Infoprint creates a temporary file for the transform output. Infoprint uses the temporary file name for the job name.

SpoolID
The job ID, which is an integer. Infoprint Manager sets the **SpoolID**.

PrinterName
The name of the Infoprint destination.

hab This function is not used.

Copy Indicates which copy is associated with this call to the exit. Initially set to 1, it increments by one each time the exit is called.

DataOffset
Indicates the offset into the source file.

DataSize
Indicates the number of bytes in the buffer.

DataPointer
Points to a buffer containing the incoming print data stream.

DataType
Indicates the type of source data the exit receives.

Valid values are:

0	Advanced Function Presentation (AFP) data stream.
1	ASCII data.

Ouput Data User-Exit Program Inputs and Outputs

The Infoprint output data user-exit program is used to monitor the outgoing print data stream from Infoprint. This exit is called at the end of the job before the error messages and the trailer page.

The source code (in the C programming language) for the sample input data user-exit program listed in “Supported types of PSF DSS user-exits” on page 1 is in the `install_path\exits\ainuxout.c` file. This sample program consists of a return code and performs no function.

The data structures for the output data user-exit program are included with the source code in the `install_path\exits\ainuexit.h` file. The code for these structures is shown in “Structure of a User-Exit Program” on page 13.

The declaration of this exit is: `void OUTDATA (OUTDATA_EXITDATA \exitdata)`

The **OUTDATA_EXITDATA** input/output parameter contains all the input and output data needed to communicate between Infoprint and the user-exit program.

To build the executable program, you must use the Microsoft Visual C++ Program, Version 5.0 pull-down menu.

The **OUTDATA_EXITDATA** structure includes the following fields.

Fields that provide input information

UserID
The system user ID of the person who submitted the job.

NodeID
The name of the system (host). The backend program sets the NodeID.

JobName
Identifies the name of the job. This is the same as the file name in the job.

Job submitters can also specify the job name with the Infoprint job attribute **job-name=name** when they submit a job. If the job name is not specified for files defined as PCL, PostScript, line data, unformatted ASCII, double-byte character set (DBCS) ASCII, or PDF, files when the job is submitted, Infoprint creates a temporary file for the transform output. Infoprint uses the temporary file name for the job name.

SpoolID
The job ID, which is an integer. Infoprint sets the **SpoolID**.

PrinterName
The name of the Infoprint destination.

hab This function is not used.

Copy Indicates which copy is associated with this call to the exit. Initially set to 1, it increments by one each time the exit is called.

DataSize
Indicates the number of bytes in the buffer.

DataPointer
Points to a buffer containing the incoming print data stream.

Note: The output information fields used by some of the sample programs ("Other Input Values" on page 5) are available through the alternate sample user exits. Infoprint does not provide an alternate Output Data User Exit.

Structure of a User-Exit Program

All input and output variables used in Infoprint user exits are defined through either of the following two files:

- `*install_path*\exits\ainuexit.h` or
- `*install_path*\exits\ainurpt.h`

The `ainuexit.h` file contains definitions for the header, trailer, separator, accounting and audit exits, while the `ainuprt.h` file formats data for the following executable reporting utilities that apply to both the accounting and audit exits:

- Accounting Log – (**ainurpt1**; **ainurpt2**; **ainurpt3**)
- Audit Log – (**ainurpt4**; **ainurpt5**; **ainurpt6**)

These user-exit program structure files are written in the C programming language. The declarations and statements in these files show the structure of the Infoprint user-exit programs, which are imbedded as part of the user-exit program.

User-exit programs for the line data transform

Infoprint provides several sample user-exit programs for the line-data transform. Use of the user exits is optional. You specify the names of the exit programs with the **inpexit**, **indxexit**, **outexit**, and **resexit** keywords.

Infoprint provides the following sample programs:

*install_path***\exits\acif\apkinp.c**
Input-record user exit

*install_path***\exits\acif\apkout.c**
Output-record user exit

*install_path***\exits\acif\apkres.c**
Resource exit

In addition, Infoprint provides the following input-record user exits programs to translate input data streams:

- apka2e
- asciinp.c
- asciinpe.c

apka2e

*install_path***\exits\acif\apka2e.c**
Converts ASCII stream data to EBCDIC stream data.

The **apka2e** input-record exit program translates data that is encoded in ASCII (code set IBM-850) into EBCDIC (code set IBM-037) encoded data. You should use this exit when your job requires fonts such as GT12, which has only EBCDIC code points defined.

When you apply PTF UR52401 (APAR IR43438), this program also converts the ASCII code page file (ibm-850) into a corresponding EBCDIC file (ibm-037) referring to files that reside in the `install_path\uconvtab\` directory. You can use the Microsoft Visual C++ Program, Version 5.0 and edit this exit to convert any code page from ASCII to EBCDIC. To perform this conversion from a Command Prompt window, see "Using the iconv command to convert code pages" on page 21.

To execute the **apka2e** input record exit program, set the following keywords and values in your line-data transform keyword file. The **line2afp parmdd** keyword identifies the keyword file.

```
inpexit=install_path\bin\apka2e
cc=yes
cctype=z
```

asciinp.c

*install_path***\exits\acif\asciinp.c**
: Converts unformatted ASCII data that contains carriage returns and form feeds into a record format that contains an American National Standards Institute (ANSI) carriage control character. This exit encodes the ANSI carriage control character in byte 0 of every record.

The **asciinp** input-record exit program transforms an ASCII data stream into a record format that contains a carriage control character in byte 0 of every record. If byte 0 of the input record is an ASCII carriage return (X'0D'), byte 0 is transformed into an ASCII space (X'20') that causes a data stream to return and advance one line; no character is inserted. If byte 0 of the input record is an ASCII form feed character (X'0C'), byte 0 is transformed into an ANSI skip to channel 1 command (X'31') that serves as a form feed in the carriage-control byte.

To execute the **asciinp** input record exit program, set the following keywords in your line-data transform keyword file. The **line2afp parmdd** keyword identifies the keyword file:

```
inpexit=install_path\bin\asciinp
cc=yes
cctype=z
```

asciinpe.c

*install_path***\exits\acif\asciinpe.c**
: Converts unformatted ASCII data into a record format just as **asciinp.c** does, and then converts the ASCII stream data to EBCDIC stream data.

The **asciinpe** input-record exit program combines both of the user input-record exits described above. To execute, specify `inpexit=install_path\bin\asciinpe` as the exit program in the keyword file and follow the directions specified for both **apka2e** and **asciinp**.

When you apply PTF UR52401 (APAR IR43438), this program also converts the ASCII code page file (ibm-850) into a corresponding EBCDIC file (ibm-037) referring to files that reside in the `install_path\uconvtab\` directory. You can use the Microsoft Visual C++ Program, Version 5.0 and edit this exit to convert any code page from ASCII to EBCDIC. To perform this conversion from a Command Prompt window, refer to "Using the iconv command to convert code pages" on page 21.

While the **asciinp** and **asciinpe** input-record exits do not recognize other ASCII printer commands, you can modify these exits to account for the following:

- backspacing (X'08')
- horizontal tabs (X'09')
- vertical tabs (X'0B')

In addition to the information above, refer to the prolog of the **asciinp.c** source file that is provided with Infoprint Manager in the *install_path*\exits\acif directory for more information on using and modifying these programs.

The C language header file for all ACIF exit programs is provided in *install_path*\exits\acif\apkexits.h along with the build rules for the ACIF user exits in *install_path*\exits\acif\Makefile.

To build the executable program, you must use the Microsoft Visual C++ Program, Version 5.0 pull-down menu.

For more information about compiling user-exit programs, refer to "Compiling and Installing the User-Exit Program" on page 3.

Input Record Exit

The line-data transform provides an exit that enables you to add, delete, or modify records in the line-data input file. The program invoked at this exit is defined by the value of the **inpexit** keyword of the **line2afp** command.

This exit is called after each record is read from the input file. The exit can request that the record be discarded, processed, or processed with control returned to the exit for the next input record. The largest record that Infoprint Manager can process is 32756 bytes. This exit is not called when the line-data transform is processing resources from directories.

The following example provides a sample C language header that describes the control block that is passed to the exit program:

```
typedef struct_INPEXIT_PARMS /* Input record exit Parameters */
{
char *work; /*Address of 16-byte static work area */
PFATTR *pfattr; /*Address of print file attribute information */
char record; /*Address of the input record */
void reserved1; /*Reserved for future use */
unsigned short recordln; /*Length of the input record */
unsigned short reserved2; /*Reserved for future use */
char *work; /*Add, delete or process the record*/
char *eof; /*EOF indicator */
}
INPEXIT_PARMS
```

The address of the control block containing the following parameters is passed to the input record exit:

work (Bytes 1–4)
A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. The user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)
A pointer to the print file attribute data structure. Refer to "Attributes of the Line Data Input File" on page 20 for more information on the format of this data structure and the information it contains.

record (Bytes 9–12)
A pointer to the first byte of the input record including the carriage control character. The record resides in a buffer that resides in storage allocated by the line-data transform but the exit program is allowed to modify the input record.

reserved1 (Bytes 13–16)
These bytes are reserved for future use.

recordln (Bytes 17–18)
Specifies the number of bytes (length) of the input record. If the input record is modified, this parameter must also be updated to reflect the actual length of the record.

reserved2 (Bytes 19–20)
These bytes are reserved for future use.

request (Byte 21)
Specifies how the line-data transform processes the record. On entry to the exit program, this parameter is X'00'. When the exit program returns control to line-data transform, this parameter must have the value X'00', X'01', or X'02', where:

X'00'	Specifies that the line-data transform should process the record.
X'01'	Specifies that the line-data transform should not process the record.
X'02'	Specifies that the line-data transform should process the record and then return control to the exit program to allow it to insert the next record. The exit program can set this value to save the current record, insert a record, and then supply the saved record at the next call. After the exit inserts the last record, the exit program must reset the request byte to X'00'.

A value of X'00' on entry to the exit program specifies that the record be processed. If you want to ignore the record, change the request byte value to X'01'. If you want the record to be processed, and you want to insert an additional record, change the request byte value to X'02'. Any value greater than X'02' is interpreted as X'00', and the exit processes the record.

Note: Only one record can reside in the buffer at any time.

eof (Byte 22)
An end-of-file (eof) indicator. This indicator is a one-byte character code that specifies whether an eof condition has been encountered.

When eof is signaled (`eof value='Y'`), the last record has already been presented to the input exit, and the input file has been closed. The pointer record is no longer valid. Records may not be inserted when eof is signaled. The following are the only valid values for this parameter:

Y	**Specifies that eof has been encountered.**
N	**Specifies that eof has not been encountered.**

This end-of-file indicator allows the exit program to perform some additional processing at the end of the file. The exit program cannot change this parameter.

Output Record Exit

Using the output-record exit, you can modify or ignore the records the line-data transform writes into the output file. The program invoked at this exit is defined by the outexit keyword of the **line2afp** command.

The exit receives control before a record (structured field) is written to the output document file. The exit can request that the record be ignored or processed. The largest record that the exit can process is 32752 bytes, not including the record descriptor word. The exit is not called when the line-data transform is processing resources.

The following example contains a sample C language header that describes the control block passed to the exit program:

```
typedef struct_OUTEXIT_PARMS /* Output record exit Parameters */
{
char *work; /*Address of 16-byte static work area*/
PFATTR *pfattr; /*Address of print file attribute information */
char *record; /*Address of the output record*/
void reserved1; /*Reserved for future use */
unsigned short recordln; /*Length of the input record */
char request; /*Add, delete or process the record */
char *eof; /*EOF indicator */
}
OUTEXIT_PARMS
```

The address of the control block containing the following parameters is passed to the output record exit:

work (Bytes 1–4)
A pointer to a static, 16-byte memory block. The exit program can use this parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. A user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)
A pointer to the print file attribute data structure. Refer to "Attributes of the Line Data Input File" on page 20 for more information on the format of this data structure and the information it contains.

record (Bytes 9–12)
A pointer to the first byte of the output record. The record resides in a 32KB (where KB equals 1024 bytes) buffer. The buffer resides in storage allocated by the line-data transform, but the exit program is allowed to modify the output record.

recordln (Bytes 13–14)
Specifies the length, in bytes, of the output record. If the output record is modified, this parameter must also be updated to reflect the actual length of the record.

request (Byte 15)
Specifies how the line-data transform processes the record. On entry to the exit program, this parameter is X'00'. When the exit program returns control to the line-data transform, this parameter must have the value X'00' or X'01', where:

X'00'	Specifies that the line-data transform should process the record.
X'01'	Specifies that the line-data transform should not process the record.

A value of X'00' on entry to the exit program specifies that the record be processed. If you want to ignore the record, change the request byte value to X'01'. Any value greater than X'01' is interpreted as X'00'; the exit processes the record.

Note: Only one record can reside in the buffer at any time.

eof(Byte 16)
An end-of-file (eof) indicator. This indicator is a one-byte character code that signals when the line-data transform has finished writing the output file.

When eof is signaled (`eof value='Y'`), the last record has already been presented to the output exit. The pointer record is no longer valid. Records may not be inserted after eof is signaled. The following are the only valid values for this parameter:

Y	Specifies that the last record has been written.
N	Specifies that the last record has not been written.

This end-of-file flag, used as a last-call indicator, allows the exit program to return control to the line-data transform. The exit program cannot change this parameter.

Resource Exit

The line-data transform provides an exit that enables you to filter (exclude) resources. This exit is useful in controlling resources at the file-name level. For example, assume you only wanted to use those fonts that are not shipped with Infoprint Manager. You could code this exit program to contain a table of all fonts shipped with Infoprint and filter those from the resource file. Security is another consideration for using this exit because you could prevent certain named resources from being included. The program invoked at this exit is defined by the resexit keyword of the **line2afp** command.

This exit receives control before a resource is read from a directory. The exit program can request that the resource be processed or ignored (skipped), but it cannot substitute another resource name in place of the requested one. If the exit requests that any overlay be ignored, the line-data transform automatically ignores any resources the overlay may have referenced (that is, fonts and page segments).

The following example contains a sample C language header that describes the control block passed to the exit program:

```
typedef struct_RESEXIT_PARMS /* Resource record exit Parameters */
{
char *work; /*Address of 16-byte static work area */
PFATTR *pfattr; /*Address of print file attribute information */
char *resname[8]; /*Name of the requested resource */
char restype; /*Type of requested resource */
char request; /*Ignore or process the request */
char *eof; /*Last call indicator */
}
RESEXIT_PARMS
```

The address of the control block containing the following parameters is passed to the resource record exit:

work (Bytes 1–4)
A pointer to a static, 16-byte memory block. The exit program can use this

parameter to save information across calls (for example, pointers to work areas). The 16-byte work area is aligned on a full word boundary and is initialized to binary zeros prior to the first call. A user-written exit program must provide the code required to manage this work area.

pfattr (Bytes 5–8)
A pointer to the print file attribute data structure. Refer to "Attributes of the Line Data Input File" on page 20 for more information on the format of this data structure and the information presented.

resname (Bytes 9–16)
Specifies the name of the requested resource. The exit program cannot modify or change this value.

restype (Byte 17)
Specifies the type of resource the name refers to. This is a one-byte hexadecimal value where:

X'03'	Specifies a GOCA (graphics) object.
X'05'	Specifies a BCOCA (barcode) object.
X'06'	Specifies an IOCA (IO image) objec.t
X'40'	Specifies a font character set.
X'41'	Specifies a code page.
X'FB'	Specifies a page segment.
X'FC'	Specifies an overlay.

The line-data transform does not call this exit for the following resource types:

Page definition
The page definition (**pagedef** keyword) is a required resource for transforming a line-data file.

Form definition
The form definition (**formdef** keyword) is a required resource for printing a transformed line-data file.

Coded fonts
The line-data transform must process coded fonts to determine the names of the code pages and font character sets they reference. This is necessary in creating Map Coded Font-2 (**MCF-2**) structured fields.

request (Byte 18)
Specifies how the line-data transform processes the resource. On entry to the exit program, this parameter is X'00'. When the exit program returns control to the line-data transform, this parameter must have the value X'00' or X'01' where:

X'00'	Specifies that the line-data transform should process the resource.
X'01'	Specifies that the line-data transform should ignore the resource.

A value of **X'00'** on entry to the exit program specifies that the resource be processed. If you want to ignore the resource, change the request byte value to **X'01'**. Any value greater than **X'01'** is interpreted as **X'00'** and the resource is processed.

eof (Byte 19)

An end-of-file (**eof**) indicator. This indicator is a one-byte character code that signals when the line-data transform has written the last record.

When **eof** is signaled (`eof value = 'Y'`), the last record has already been presented to the resource exit. The pointer **record** is no longer valid. Records may not be inserted after **eof** is signaled. The following are the only valid values for this parameter:

Y	Specifies that the last record has been written.
N	Specifies that the last record has not been written.

This end-of-file flag, used as a last-call indicator, returns control to the line-data transform. The exit program cannot change this parameter.

Non-Zero Return Codes

If the line-data transform receives a non-zero return code from any exit program, the line-data transform issues message 0425-412 and terminates processing.

Attributes of the Line Data Input File

Note: This data structure is provided for informational purposes only.

The line-data transform provides information about the attributes of the line-data input file in a data structure available to the line-data transform user exits. The following example shows the format of this data structure.

```
typedef struct_PFATTTR /* Print File Attributes */
{
char cc[3]; /*Carriage controls? – "YES" or "NO" */
char cctype[1]; /*Carriage control type - A(ANSI), M (Machine), Z (ASCII) */
char chars[20]; /*CHARS values, including commas (eg.. GT12,GT15) */
char formdef[8]; /*Form Definition (FORMDEF) */
char pagedef[8]; /*Page Definition (PAGEDEF) */
char prmode[8]; /*Processing mode */
char trc[3]; /*Table Reference Characters – "YES" or "NO" */
}
PFATTR;
```

The address of the control block containing the following parameters is passed to the user exits:

cc (Bytes 1–3)

The value of the **cc** keyword as specified with the **line2afp** command. The line-data transform uses the default value (**yes**) if this keyword is not explicitly specified.

cctype (Byte 4)

The value of the **cctype** keyword as specified with the **line2afp** command. The line-data transform uses the default value (**z**) for ANSI carriage-control characters that are encoded in ASCII, if this keyword is not explicitly specified.

chars (Bytes 5–24)

The value of the **chars** keyword as specified with the **line2afp** command, including any commas that separate multiple font specifications. Because the **chars** keyword has no default value, this field contains blanks if no values are specified.

formdef (Bytes 25–32)
The value of the **formdef** keyword as specified with the **line2afp** command. Because the **formdef** keyword has no default value, this field contains blanks if no value is specified.

pagedef (Bytes 33–40)
The value of the **pagedef** keyword as specified with the **line2afp** command. Because the **pagedef** keyword has no default value, this field contains blanks if no value is specified.

prmode (Bytes 41–48)
The value of the **prmode** keyword as specified with the **line2afp** command. Because the **prmode** keyword has no default value, this field contains blanks if no value is specified.

trc (Bytes 49–51)
The value of the **trc** keyword as specified with the **line2afp** command. The line-data transform uses the default value (**no**) if this keyword is not explicitly specified.

Using the iconv command to convert code pages

From a Command Prompt window, you can invoke the **iconv** command to convert a code page from one format to another (for example, to convert ASCII to EBCDIC). This command is useful when the line data code pages must be changed so they match the code pages for the available font resources for a job. These are often applied for job resources (such as page definitions and form definitions) that have been created on either an AIX or an MVS operating system and sent to Windows NT or Windows 2000.

Use the following syntax to invoke this command:

```
iconv -f nnn -t nnn infile [> outfile]
```

-f
The existing code page that needs to be changed, using the format of *nnn* where *nnn* represents the particular code page. The space between the flag and its value is optional.

-t
The code page that is the result of the command, using the format of *nnn* where *nnn* represents a three-to-five number alphanumeric code page name in the format -*nnn*, *nnnn*, or *nnnnn*. The space between the flag and its value is optional.

infile
The input text file name to which you want to convert. If you do not specify a file name, the **iconv** command expects input from standard input (**STDIN**).

outfile
The output file where the result of the **iconv** transform is stored. Use an output file to pass the **iconv** output to a customized program designed to process this output. If you do not specify a file name, the output goes to standard out (**STDOUT**).

Example 1

To convert an ASCII code page to an EBCDIC code page, open a Command Prompt window and type the following:

```
iconv -f 850 -t 500 input.txt > output.txt
```

Example 2

To convert a Japanese ASCII code page to a Japanese EBCDIC code page, open a Command Prompt window and type the following:

```
iconv -f 932 -t 939 input_jpn.txt > output_jpn.txt
```

Back to Administrator procedures