

This section details the Banner Processing Output Pre-processing Module that can be optionally integrated into certain Tasks used by TN Interfaces.

The following information is included:

	PAGE
Overview	21.2
Operation	21.3
Configuration	21.4
Parameter Files	21.5
Defining an Identification String (STRING)	21.7
Determining Search Lines (BANNER / IDENTIFY)	21.9
Determining the Action to be Taken (ACTION / JOBSPLIT)	21.13
Defining System Fields	21.18
PCL Data	21.22
Processing IBM Channel Data and CCWs	21.25
Dumping	21.27
Example Process	21.28

Overview

Banner Processing is an Output Pre-processing Module, that can be optionally integrated into all job based Output Modules. It is used to search for banner, job separation, trailer and any other identifiable page within the data received, and when found perform one of the following actions:

- Split the data into separate jobs, before or after the identified page.
- Associate a separated job with a Printer Number, allowing Configuration Parameters appended with the same number to be used for this job alone.
- Delete banner, job separation, trailer and any other identified page from the data.
- Hold and release separated jobs (if a Spooling System is used).
- Copy identified variable data to System Fields.

Operation

Banner Processing receives data from the Input Module in the Internal Format, line by line and page by page. Each page is determined by both logical and actual top-of-form indicators that also acts as a reset for the internal line counter.

Character strings, that may appear on certain lines on certain pages within the data received by Banner Processing, are specified as Identification Strings within a user defined Parameter File. The line numbers where they are likely to appear, along with the action to be taken when a page is identified, must also be referenced within this file.

NOTE: In order to identify a page, a single or 'group' of Identification Strings, together with the line numbers where they will be searched, will need to be defined. To help ensure that the correct page is identified, it is normal to define 'groups' of Identification Strings - only when all of them have been found, will a page be identified. A single Identification String can be used as long as it does not appear on the searched lines on any other page within the data.

As each line is received by Banner Processing it is counted and buffered, and when the first line, defined within the Parameter File, is reached, it is checked for the occurrence of an associated Identification String. This continues until each of the lines referenced has been checked on the page or until the page is identified.

Pages that are not identified are passed to the Output Module, the internal line counter reset and the whole process started again for the new page. This continues until a page is identified or the end of the data is reached.

When a page is identified, an action that is associated with it is performed. When complete, Banner Processing continues to process the remaining pages, until another page is identified or the end of the data is reached.

NOTE: Banner Processing advances through lines and pages within data sequentially and is unable to 'move' or 'jump' back to lines already processed or pages already passed to the Output Module.

Configuration

Banner Processing is configured by applying Configuration Parameters to an Output Module within an appropriate Task Configuration File.

For further details, see Section 5, Task Configuration Files.

Configuration Parameters

The following tables detail the Configuration Parameters and their values that are used for configuration, with a description of each provided alphabetically within Section 29, Configuration Parameters.

The Configuration Parameters listed are categorised as follows:

- Installation
- Dumping

Installation

Parameter	Values	Default
JOBSPLIT	BANNER.xxx	

Dumping

Parameter	Values	Default
DUMPJOBSPLIT *	ALL NO YES	•

* The DUMP parameter must also be enabled for the DUMPJOBSPLIT parameter to work. For further details, refer to Dumping, below.

Parameter Files

Banner Processing uses a standard ASCII Parameter File to define its search and processing criteria. The file used is determined by the filename set for the JOBSPLIT parameter in an associated Task Configuration File.

Location

The location of Banner Processing Parameter Files will depend upon the TN Interface being used:

- **TN3000** A:\
- **TN4000** \PRINTQ
- **TN5000** \SPOOLQ
- **TN6000** \TN6000
- **TN8000** \SpoolQ NT\CONFIG

Creating and Editing

Parameter Files will sometimes require modification and new ones may need to be created. How this is done will depend upon what operating system the TN Interface is running under.

CAUTION: If the file is to be edited, it is advisable to make a copy before any amendments are made.

For further details about creating and editing user definable files such as Parameter Files, see Section 2, Introduction - Creating and Editing User Definable Files.

Structure

Parameter Files (Fig. 21.1) are normally structured in a similar way. They contain commands at the top to define Identification Strings (STRINGnn), followed by others that determine the line(s) on a page where each of the Identification Strings may be found (BANNERnn or IDENTIFYnn), and a command to perform a certain action associated with an identified page (JOBSPLITnn or ACTIONnn).

Fig. 21.1
Example Parameter File

```
STRING1=**START*
STRING2=**END*
STRING3=**START1
STRING4=**END1
STRING5=?-17-??-10#1-?
IDENTIFY1=1@28-35&5@40
ACTION1=BEFORE,MULTI1,PRINTER1
IDENTIFY2=2@28-35
ACTION2=AFTER,MULTI1
IDENTIFY3=3@28-35&5@40
ACTION3=BEFORE,MULTI1,PRINTER2
IDENTIFY4=4@28-35
ACTION4=AFTER,MULTI1
```

NOTE: Parameter Files will operate when structured in any manner. It is good practice however, to define them in a similar way to the above example, so that they can be easily read.

Remarks

Remarks can be added to Parameter Files to provide a title and to help identify commands used. They can be added by including either the REM statement or \

before the required remark. For example:

```
REM This identifies banner page 1
\ This identifies banner page 1
```

For early versions of Banner Processing the command `STRING99=` was used to add remarks. For example:

```
STRING99=This identifies banner page 1
```

Commands

The following commands can be used within a Parameter File:

Command	Description
ACTIONnn	Same as <code>JOBSPLITnn</code> .
BANNERnn	Determines which lines on a page Banner Processing should search for associated Identification Strings. For further details, refer to Determining Search Lines (<code>BANNER / IDENTIFY</code>), below.
CCWxx	Determines what the character <code>xx</code> in the CCW Action Translate Table should be modified to. For further details, refer to Identifying CCWs, below.
IDENTIFYnn	Same as <code>BANNERnn</code> .
JOBSPLITnn	Determines the action to be taken by Banner Processing when a page is identified. For further details, refer to Determining the Action to be Taken (<code>ACTION / JOBSPLIT</code>), below.
PROCESS	Determines whether to process CCWs and data in the Channel Tunnel format. For further details, refer to Identifying CCWs, below.
STRINGnn	Defines an Identification String that is used by the <code>BANNERnn</code> or <code>IDENTIFYnn</code> commands to identify a line on a page. For further details, refer to Defining an Identification String (<code>STRING</code>), below. It is also used to determine the System Fields and the variable data that is to be copied to them. For further details, refer to System Fields, below.

Each command that is followed by a number (nn), is referenced by another command.

NOTE: All commands must be entered in uppercase.

Defining an Identification String (STRING)

STRINGnn is the command used to define an Identification String that the BANNERnn or IDENTIFYnn commands use to identify a line on a page.

The command is also used to determine the System Fields and the variable data to be copied to them. For details, refer to System Fields, below.

NOTE: Upto 100 STRINGnn commands can be used within a Parameter File.

Format

STRINGnn has the following format:

```
STRINGnn=xxxxx
```

Identification Number (nn)

The value nn is a number between 1 and 100 that is referenced by the BANNERnn or IDENTIFYnn commands.

Identification String (xxxxx)

The value xxxx refers to an Identification String, upto 160 characters in length, representing the format of a line that appears on a page.

EXAMPLE: As a very simple example, if ALPHA appeared as the first five characters on a line within a page, and the rest of the line is ignored, the following command would be used to define the Identification String:

```
STRING1=ALPHA
```

NOTE: Banner Processing assumes that the Identification String defined by the STRINGnn command starts at the left-most position of a line (column 1).

NOTE: Banner processing assumes that all lines contain any number of trailing spaces. This means that an Identification String can end with a space whether one is present or not.

Wildcard Statements

The following table indicates the wildcard statements that can be used to represent variable data within an Identification String.

Wildcard	Description
?	Any single variable character. The question mark (?) character cannot be represented.
?-?	Variable data of an unknown length.
?-nn-?	Variable data with a fixed length of nn characters.
?<>?	Variable data of an unknown length.
?<mm>?	Variable data of an unknown length upto a maximum of mm characters.
?<nn-mm>?	Variable data with a minimum length of nn characters and a maximum length of mm characters.
?--?	Variable data of an unknown length.

NOTE: The maximum number of characters that can be determined by the above wildcard statements (except ?) is 127.

Multiple Fixed and Variable Data Statements

Where a line to be identified consists of a sequence of fixed and variable data, the STRINGnn command can be defined in order (starting from column one),

with statements to represent this. Any sequence of fixed and variable data statements can be used by the command.

EXAMPLE: The following command determines multiple fixed and variable data on a single line:

```
STRING1=?-8-?ALPHA?-?BETA
```

The `STRING1` command above is used to identify a line where the first eight characters are variable, the following five defined as fixed data `ALPHA`, the next a variable and unknown number of characters upto fixed data `BETA`, after which there maybe any number of characters on the line.

NOTE: The end of a `STRING` command is assumed to logically be the variable data statement `?-?`, that represents all other characters on the line.

NOTE: `STRINGnn` commands can be used any number of times within the `BANNERnn` or `IDENTIFYnn` commands.

Determining Search Lines (BANNER / IDENTIFY)

The BANNERnn and IDENTIFYnn commands are synonymous and are used to define which line(s) on a page should be searched for associated Identification Strings in order to identify a page. If all of the Identification Strings are found for a particular 'group', Banner Processing carries out a corresponding JOBSPLIT or ACTION command.

NOTE: Banner Processing uses an internal counter to count each line on a page. The start of a page is defined and the line counter reset when a logical or actual internal top-of-form is encountered.

NOTE: References and examples that follow refer to the IDENTIFYnn command.

Format

IDENTIFYnn has the following format:

```
IDENTIFYnn=STRINGnn@LINESaa-bb
```

Identification Number (nn)

The nn value is an identification number between 1 and 100 that is referenced by the JOBSPLIT or ACTION commands.

EXAMPLE: The following command:

```
IDENTIFY1
```

is referenced by:

```
ACTION1
```

```
JOBSPLIT1
```

Character String (STRINGnn)

The STRINGnn statement is used to reference the specified Identification String that is to be searched for on a line.

NOTE: The STRING statement can be omitted and just the identification number (nn) used in its place.

EXAMPLE: The outcome of the following commands is identical:

```
IDENTIFY1=STRING1@LINES28-35
```

```
IDENTIFY1=1@LINES28-35
```

Statement Separation Character (@)

The @ character is used to separate STRING / LINES statements or numbers.

Line Number(s) (LINESaa-bb)

The LINESaa-bb statement defines the first (aa) and last (bb) lines on a page where an Identification String may appear. These lines along with the ones that fall between will be checked for the Identification String.

EXAMPLE: The following command checks lines 28 to 35 for the Identification String defined by STRING1:

```
IDENTIFY1=STRING1@LINES28-35
```

NOTE: The lines referenced must be sequential, because Banner Processing cannot interrogate lines already processed.

EXAMPLE: The following **cannot** be used:

```
IDENTIFY1=STRING1@LINES35-28
```

If only one line is to be defined, only a single number without a hyphen should

be used.

EXAMPLE: The following command searches line 28 for the Identification String defined by STRING1:

```
IDENTIFY1=STRING1@LINES28
```

NOTE: The maximum number of lines that Banner Processing can count for a single IDENTIFY command is 200.

NOTE: The LINES statement can be omitted and just the identification number(s) used in its place.

EXAMPLE: The outcome of the following commands is identical:

```
IDENTIFY1=STRING1@LINES28-35
```

```
IDENTIFY1=STRING1@28-35
```

```
IDENTIFY1=1@28-35
```

As are the following commands:

```
IDENTIFY1=STRING1@LINES28
```

```
IDENTIFY1=STRING1@28
```

```
IDENTIFY1=1@28
```

Defining Multiple Statements (&)

Due to the likelihood that a single Identification String could appear on a number of pages within data, it will normally be necessary to define multiple or 'groups' of statements so that the correct page is identified.

NOTE: All Identification Strings referenced within a single IDENTIFY command must be found on a page to identify it.

Each STRING and LINES statement is separated by the & character.

EXAMPLE: The following command would be used to search for an Identification String defined by STRING1 on lines 28 to 35, and another defined by STRING2 on lines 45 to 49, of the same page:

```
IDENTIFY1=STRING1@LINES28-35&STRING2@LINES45-49
```

Each of the lines referenced by each statement must not precede or overlap lines referenced by previous statements for the same IDENTIFY command.

EXAMPLE: The following **cannot** be used:

```
IDENTIFY1=STRING1@LINES28-35&STRING2@LINES25-29
```

Searching Multiple Contiguous Pages (+)

The + character can be included within an IDENTIFYnn command to instruct Banner Processing to search for Identification Strings over multiple contiguous pages. This is necessary because certain data sent by some Sources may contain a number of contiguous pages that identify a job.

EXAMPLE: The following command will firstly search for an Identification String associated with STRING1 on lines 28 to 35 of one page, followed by STRING2 on lines 30 to 40 on the next page.

```
IDENTIFY1=STRING1@LINES28-35+STRING2@LINES30-40
```

NOTE: All Identification Strings must be found across multiple contiguous pages to identify the 'page'.

NOTE: Although the line number is reset to 1 for each consecutive page, the maximum number of lines that can be searched is 200.

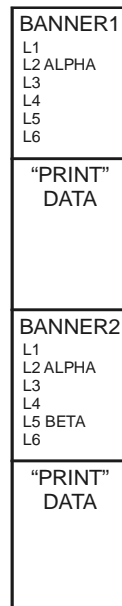
EXAMPLE: If lines 1 to 50 are searched over multiple pages, upto four contiguous pages can be searched.

For details about the other use of the + delimiter, refer to Identifying the Correct Page (+), below.

Identifying the Correct Page (+)

If the data received by Banner Processing is likely to contain the same Identification String(s) on the same line(s) on multiple pages (Fig. 21.2), it is important to ensure that the correct page is identified by an IDENTIFY command.

Fig. 21.2
Data containing banner pages
with the same Identification
String (ALPHA)



EXAMPLE: The following commands relate to the diagram (Fig. 21.2) above:

```
STRING1=ALPHA
STRING2=BETA
IDENTIFY1=1@2
IDENTIFY2=1@2&2@5
```

With reference to the commands above, the IDENTIFY1 command will identify **both** the BANNER1 and BANNER2 pages. The IDENTIFY2 command will never be satisfied because the page will be identified by IDENTIFY1 using the ALPHA string and an associated action performed before the line containing the BETA string was ever reached.

To overcome this problem a + delimiter is used at the end of an appropriate IDENTIFY command (i.e. IDENTIFY1 for the example above) to indicate to Banner Processing that the end of the page must be reached before the IDENTIFY1 command is satisfied and the action associated with it performed. If another IDENTIFY command is satisfied then the action associated with that is performed instead.

EXAMPLE: The following commands relate to the diagram (Fig. 21.2) above:

```
STRING1=ALPHA
STRING2=BETA
IDENTIFY1=1@2+
IDENTIFY2=1@2&2@5
```

With reference to the commands above, the IDENTIFY1 and IDENTIFY2 commands locate the ALPHA string on line 2 of the BANNER1 page. The + delimiter at the end of the IDENTIFY1 command ensures that the end of the page must be reached before the IDENTIFY1 command is satisfied, therefore ensuring that the action associated with it is performed. The BANNER1 page does not contain the BETA string so IDENTIFY2 cannot be satisfied, indicating to Banner Processing that the action associated with IDENTIFY1 is performed.

On the BANNER2 page, both the IDENTIFY1 and IDENTIFY2 commands locate the ALPHA string on line 2. The + delimiter at the end of the IDENTIFY1 command ensures that the end of the page must be reached for it to be satisfied. Prior to this, the IDENTIFY2 command finds the BETA string on line 5, so both its page identification criteria are satisfied. The action associated with IDENTIFY2 is therefore performed as the IDENTIFY1 command is yet to be fully satisfied.

Ensuring Efficiency

To ensure that Banner Processing operates as efficiently as possible, it is advised that the first Identification String defined for a page appears near the top of the page. This will ensure that the page is released quickly if the Identification String is not found. As multiple statements within an IDENTIFYnn command can be used, this early check need not be fully unique to represent the page, but only need act as an initial 'filter' dismissing the majority of pages.

If the first or only identified string is located on a line near the bottom of a page, a large amount of data has to be buffered before it can be found. This may result in performance loss.

Determining the Action to be Taken (ACTION / JOBSPLIT)

The ACTIONnn and JOBSPLITnn commands are synonymous and are used to define the action to be taken when a page is identified. When Banner Processing locates a page using the BANNERnn or IDENTIFYnn commands, a corresponding ACTIONnn or JOBSPLITnn command is actioned.

NOTE: Although both commands operate in the same way, each can be used within a Parameter File to clarify different points.

EXAMPLE: Use JOBSPLIT to split the data into jobs and ACTION where only the extraction of variable data to System Fields is required. For further details, refer to Defining System Fields, below.

NOTE: References and examples that follow refer to the ACTIONnn command.

Format

ACTIONnn has the following format:

```
ACTIONnn=[ position ] , MULTImm , PRINTERnn , [ task ]
```

Apart from [position], the statements can be in any order.

Identification Number (nn)

The value nn is a number between 1 and 100 that corresponds to the identification number given to a BANNERnn or IDENTIFYnn command.

EXAMPLE: The following command:

```
ACTION1
```

performs an action on:

```
BANNER1
```

```
IDENTIFY1
```

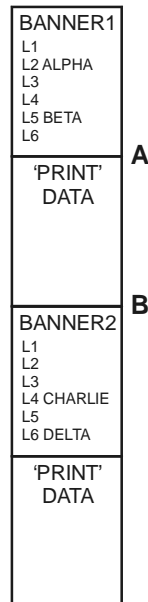
Data Split ([position])

The [position] statement determines how or whether the data is to be split when a page is located. This provides the ability to create separate jobs from a single piece of data and/or copy variable data from the identified page to System Fields.

The statements that can be applied are as follows:

Value	Description
AFTER	Splits the data after the identified page. If a Spooling System is used and RELEASE is defined by the associated ACTION command, the job is immediately passed to a queue. For further details, refer to Data Processing [task], below.
BEFORE	Splits the data before the identified page.
INCLUDE	No splitting of the data takes place, with the identified page being included within the job. It is only used when variable data found on an identified page needs to be copied to System Fields. If a Spooling System is used then this statement will prevent the job being released in a queue until a RELEASE statement is actioned or the end of the data is reached. For further details, refer to Data Processing [task], below.

Fig. 21.3
Example structure of
incoming data



EXAMPLE: With reference to the diagram above (Fig. 21.3) the following commands could be used to define Identification Strings, search for them and split the data and create three jobs; one before point A, one between points A and B, and another after point B:

STRING1=ALPHA	Defines Identification String ALPHA.
STRING2=BETA	Defines Identification String BETA.
STRING3=CHARLIE	Defines Identification String CHARLIE.
STRING4=DELTA	Defines Identification String DELTA.
IDENTIFY1=1@2&2@5	Searches for ALPHA on line two (L2) and BETA on line five (L5).
IDENTIFY2=3@4&4@6	Searches for CHARLIE on line four (L4) and DELTA on line six (L6).
ACTION1=AFTER	Splits the data at point A.
ACTION2=BEFORE	Splits the data at point B.

Banner Processing identifies the BANNER1 page from the ALPHA string on line two and the BETA string on line five, splits the data after it and passes it to the Output Module as a job. The data between points A and B is passed to the Output Module as another job, with the end being determined when the BANNER2 page is identified by the CHARLIE string on line four and the DELTA string on line six. The data is split before this page and another job created and passed to the Output Module.

NOTE: The BANNER1 page, that is merely a banner page, could be deleted instead of being passed to the Output Module as a job. For details about how to do this, refer to Data Processing [task], below.

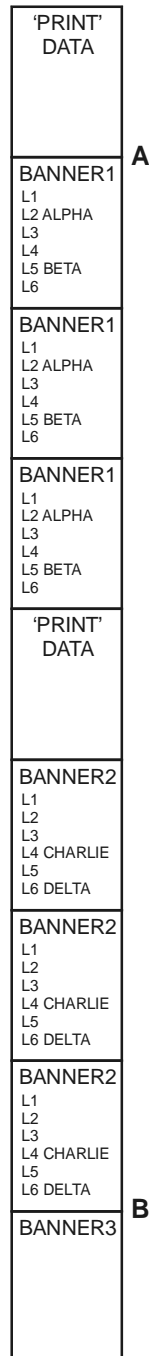
Number of Consecutive Identical Pages (MULTImm)

The MULTImm statement defines the maximum number of consecutive identical pages that can be found within data. The value of mm cannot exceed 15, and if not present, 15 is assumed.

Where multiple consecutive identical pages are identified, the effect is to include the pages within the same job.

NOTE: The term identical means that the page has been identified using the same IDENTIFY command and therefore contains the same Identification String(s) as the other page(s).

Fig. 21.4
Example structure of incoming data



EXAMPLE: With reference to the diagram above (Fig. 21.4) the following commands could be used to define Identification Strings, search for them and split the data and create a job that started at point A and ended at point B:

- STRING1=ALPHA Defines Identification String ALPHA.
- STRING2=BETA Defines Identification String BETA.
- STRING3=CHARLIE Defines Identification String CHARLIE.
- STRING4=DELTA Defines Identification String DELTA.
- IDENTIFY1=1@2&2@5 Searches for ALPHA on line two (L2) and

	BETA on line five (L5).
IDENTIFY2=3@4&4@6	Searches for CHARLIE on line four (L4) and DELTA on line six (L6).
ACTION1=BEFORE , MULTI 3	Splits the data at point A.
ACTION2=AFTER , MULTI 3	Splits the data at point B.

If only one or two consecutive identical pages were contained within the data, the command would still operate correctly.

EXAMPLE: If four identical banner pages were contained within the data, it would be split into two jobs; the first containing three banner pages, and the second containing just one, the data and three other banner pages.

NOTE: The MULTImm statement can be omitted and just a number used following the position statement.

EXAMPLE: The outcome of the following commands is identical:

```
ACTION1=BEFORE , MULTI 3
ACTION1=BEFORE 3
```

Printer Number (PRINTERnn)

The PRINTERnn statement is used to define a Printer Number, from 0 to 99, that Banner Processing associates with the job it has identified. Certain Configuration Parameters used by the Output Module can have this Printer Number appended to them so that they alone relate to the identified job. This is useful for passing jobs to particular queues within SpoolQ, determining what Destination Autorouting should output a job to, defining passwords and user names for certain Destinations, and passing values identified in specific jobs to an LPR Control File.

The Printer Number value determined for a job is held within the Printer Number (#N) System Field. If a Spooling System is used this value is written to the Start Spool File so it can be referenced whenever required.

EXAMPLE: If a TN Interface operating SpoolQ used the following Banner Processing command to split a job:

```
ACTION1=BEFORE , PRINTER2
```

And the following Configuration Parameter was defined within the Configuration File associated with the xxxxSQUO Task:

```
HOSTNAME2=Accounts
```

Then the Host Queue (#H) System Field would be defined as Accounts, resulting in the identified job being passed to the Accounts Host Queue within SpoolQ. For details about other Configuration Parameters that can be used to define System Fields, see Section 7, System Fields.

NOTE: If the PRINTERnn statement is not defined then a default of 0 is used. Jobs will therefore be associated with Configuration Parameters without an nn value appended to them (i.e. Printer Number 0).

For further details about using Printer Numbers, see Section 4, Configuration Parameter Overview.

Data Processing [task]

The task statement defines what Banner Processing is to do with a job or identified page(s).

Value	Description
HOLD	Applicable only to TN Interfaces operating a Spooling System. Gives the Start Spool File associated with a job an STH extension which holds the job until it is released by the RELEASE statement within another ACTION command. HOLD will not operate once a job has been released into a queue. For further details about holding jobs and preventing them from being placed in a queue, see Section 11, PrintQ - Spool Files.
RELEASE	Applicable only to TN Interfaces operating a Spooling System. Renames the STH extension of the Start Spool File associated with a job to STA. This indicates that the job should be passed immediately to a queue. For further details about releasing jobs into a queue, see Section 11, PrintQ - Spool Files.
DELETE	Deletes the identified page(s) from the data. Variable data that is to be copied to System Fields, is copied before the page is deleted.
ENDJOB	Provided for PCL data to determine where to end a job. For further details, see PCL Data - ENDJOB, below.

Multiple data processing tasks can be specified.

EXAMPLE: The following command will split the data after the identified page, delete the identified page and release the job into a queue of a Spooling System:

```
ACTION1=AFTER , DELETE , RELEASE
```

Defining System Fields

Banner Processing can be used to copy variable data found on an identified page to a System Field. This is achieved using the Banner Processing commands already mentioned.

For further details about System Fields, see Section 7, System Fields.

Determining the Variable Data to Copy

The STRINGnn command that is used to define Identification Strings, is also used to determine a System Field and the variable data to be copied to it.

For full details, refer to Defining an Identification String (STRING), above.

Wildcard Statements

A number of the wildcard statements that represent variable data within an Identification String, are used together with a System Field identifier, to determine what is to be copied.

The following table indicates the position of System Field identifiers within wildcard statements.

Wildcard	Description
?-#s-?	Variable data of an unknown length.
?#s#?	Variable data of an unknown length.
?-nn#s-?	Variable data with a fixed length of nn characters.
?<mm#s>	Variable data of an unknown length upto a maximum of mm characters.
?<nn-mm#s>	Variable data with a minimum length of nn and a maximum length of mm characters.

NOTE: The maximum number of characters that can be determined by any of the above wildcard statements is 127.

The value #s indicates the System Field identifier that the variable data is to be copied to. All of the System Fields, except #10, can be used, but only the number identifiers (0 to 61) can be referenced within the statement.

EXAMPLE: The following command defines variable data of ten characters in length to be copied to System Field #5:

```
STRING3=?-10#5-?
```

NOTE: For details about certain rules that should be considered when using variable data of an unknown length, refer to Increased Length System Fields, below.

Multiple Fixed and Variable Data Statements

Multiple fixed and variable data, with and without System Field identifiers, can be included in any combination within a single STRINGnn command.

EXAMPLE: The following command determines multiple fixed and variable data with and without System Field identifiers:

```
STRING1=?-8-?ALPHA?-10#5-?BETA?-#6-?[SPACE]
```

The STRING1 command above determines variable data with a fixed length of eight characters, fixed data ALPHA, variable data with a fixed length of ten characters to be copied to System Field #5, fixed data BETA, and variable data with an unknown length (terminated with a space character) that is to be copied to System Field #6.

Assigning Variable Data

The IDENTIFYnn and BANNERnn commands search for Identification Strings on specified lines within a page, and once located, assign variable data, that is associated with any System Field identifiers, to Temporary System Fields.

For full details, refer to Determining Search Lines (BANNER / IDENTIFY), above.

NOTE: References and examples that follow refer to the IDENTIFYnn command.

EXAMPLE: The following commands identify a page and assign variable data:

```
STRING1=ALPHA
STRING2=BETA
STRING3=?-10#5-?
IDENTIFY1=1@30&2@35&3@40
```

The IDENTIFY1 command searches line 30 for ALPHA, line 35 for BETA, and line 40 for variable data, and if found, assigns the variable data with a fixed length of ten characters that is found at the start of line 40 to a Temporary System Field.

EXAMPLE: The following commands identify a page and assign variable data:

```
STRING1=?-8-?ALPHA?-10#5-?BETA?-#6-?[SPACE]
IDENTIFY1=1@30-35
```

The IDENTIFY1 command searches for a line that satisfies the STRING1 command on lines 30 to 35. If found it assigns variable data with a fixed length of ten characters that is found between ALPHA and BETA, and variable data of an unknown length that follows BETA and is terminated by a space character, to Temporary System Fields.

Temporary System Fields

The IDENTIFYnn command does not copy variable data to the actual System Field defined. Instead it assigns it to a Temporary System Field that is associated with that command only. The variable data is copied to the actual System Field when an associated ACTIONnn or JOBSPLITnn command is performed.

For further details, refer to Copying Variable Data to a System Field, below.

Copying Variable Data to a System Field

When a page that contains variable data to be copied is identified, an ACTIONnn or JOBSPLITnn command associated with the IDENTIFYnn or BANNERnn command, copies the variable data assigned to Temporary System Fields to actual System Fields.

For further details, refer to Determining the Action to be Taken (ACTION / JOBSPLIT), above.

NOTE: References and examples that follow refer to the ACTIONnn command.

NOTE: Any valid ACTIONnn statement will copy the associated variable data assigned to Temporary System Fields to actual System Fields.

EXAMPLE: The following commands identify a page, assign variable data, and copy it to a System Field:

```
STRING1=ALPHA
STRING2=BETA
STRING3=?-10#5-?
IDENTIFY1=1@30&2@35&3@40
ACTION1=INCLUDE
```

The ACTION command indicates that the identified page should be included in the job and copies the variable data assigned to Temporary System Field #5 to System Field #5.

EXAMPLE: The following commands identify a page, assign variable data, and copy it to System Fields:

```
STRING1=?-8-?ALPHA?-10#5-?BETA?-#6-?[SPACE]
IDENTIFY1=1@30-35
ACTION1=AFTER
```

The ACTION1 command splits the data after the identified page and copies the variable data assigned to Temporary System Fields to System Fields #5 and #6.

Increased Length System Fields

If a System Field is used under the MS-DOS operating system, it can hold a maximum of 16 bytes of data (characters), while if it is being used under the Windows NT/2000 operating system, it can hold a maximum of 128 bytes of data (characters). For this reason it is important that the number of characters to be copied to a System Field is defined correctly.

EXAMPLE: Line 30 within a page of incoming data is defined as follows:

```
USERNAME:USER1                LOCATION:ROOM462
```

Only the user name (USER1) is required in a System Field, so a Parameter File would be defined with the following:

```
STRING1=USERNAME: ?-#1-?
IDENTIFY1=1@30
ACTION1=INCLUDE
```

Banner Processing uses USERNAME: to identify the line (and page) and copy variable data of an unknown length to System Field #1.

If a 16 byte System Field is used, the following would be copied to System Field #1:

```
USER1
```

This is because only 16 characters are copied, which in this instance happens to be USER1 followed by 11 trailing spaces. If, however, a 128 byte System Field is used, the following would be copied to System Field #1:

```
USER1                LOCATION:ROOM462
```

To prevent this from happening the following should be defined:

```
STRING1=USERNAME: ?-#1-?[SPACE]
```

The space character used at the end of the STRING1 command will indicate the end of the variable data to be copied to System Field #1. Alternatively, variable data of a fixed length could be defined in the following manner:

```
STRING1=USERNAME: ?-n#1-?
```

The value determined by n indicates the number of characters to copy to the System Field.

Using with SpoolQ

System Fields are useful for defining things such as job names, the number of copies and the queue within SpoolQ that a job should be passed to. All these values can often be obtained from banner, trailer, job separation or other identified pages found within incoming data.

NOTE: Certain System Fields that are used by SQUO, such as Host Queue Name (#17), or Copies (#12), and defined from variable data found in the middle or at the end of a job (i.e. from a trailer page), can only be used if the job is immediately held when its Start Spool File is written. If it is not held, the System Fields will

not be written to the Start Spool File because by the time the line that they appear on is identified, the job may already be being output. Holding the job, ensures that these values are written.

For details about holding a job, refer to Determining the Action to be Taken (ACTION / JOBSPLIT), above.

For further details about SpoolQ, see Section 12, SpoolQ.

PCL Data

Due to the structure of Print Command Language (PCL) data, certain issues must be considered before Banner Processing can take place.

For further details about PCL, see Section 28, Data Formats.

Resources

CAUTION: Resources, that appears at the beginning of a job within data, contain information about fonts and the form structure of that job (Fig. 21.5). It can often contain a large amount of information which cannot normally be processed by TN3000 Interfaces that buffer everything within (limited) memory. For this reason, only TN Interfaces that operate a Spooling System can reliably perform Banner Processing on PCL data. This is because the data that passes from the Input Module to the Spooling System is written to the hard drive of the TN Interface as a sequence of Spool Files, therefore allowing large amounts of information to be processed.

Preparing the Data

AutoFilter should be used to format PCL data so it can be interpreted correctly by Banner Processing. The use of this Input Pre-processing Module effectively allows Banner Processing to 'see' the normal data.

For further details about AutoFilter, see Section 20, AutoFilter.

Defining Character Strings

Even after using AutoFilter, PCL data may still have certain escape sequences held within it. If this is the case it is important to make use of wildcard statements when defining Identification Strings.

EXAMPLE: To ignore escape sequences (and other data) when locating the fixed data ALPHA, the following command should be used:

```
STRING1=?-?ALPHA
```

For further information about wildcard statements, refer to Defining an Identification String (STRING) - Format - Wildcard Statements, above.

Line Numbers

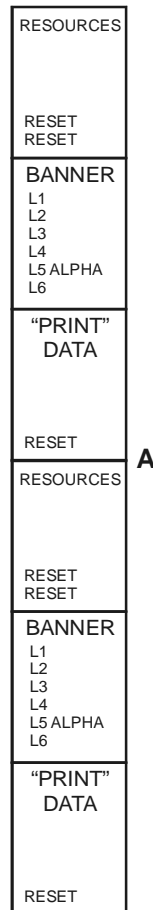
The line numbers used by the BANNERnn or IDENTIFYnn commands are the logical line numbers assigned by Banner Processing. AutoFilter logically defines any AutoFilter line movement action as a single line move.

After AutoFilter has processed the PCL data, Banner Processing can 'see' all the real and logical line movements.

Using the ENDJOB Statement

PCL jobs are normally made up of a resource section, a banner page, and the 'print' data, immediately followed by other jobs in the same format (Fig. 21.5). For this reason, data containing more than one job cannot be split using the normal BEFORE statement because resources normally precede the banner page and may include PCL resets.

Fig. 21.5
Example PCL data



To determine where to split PCL data, the ENDJOB statement is used by the commands JOBSPLITnn and ACTIONnn. The statement works by ending a job when a certain reset command is encountered, or if that fails, before a new banner page. The following statements are used to do this:

```
ENDJOB@RESET
ENDJOB@NEXTBANNER
ENDJOB@NEXTBANNER/RESET
```

NOTE: The ENDJOB statement is reset at the start of each new job.

For details on how to apply the ENDJOB statement, see Determining the Action to be Taken - Format - Data Processing ([task]), above.

ENDJOB@RESET

The ENDJOB@RESET statement ends the PCL job when the next PCL reset command is encountered. A number of reset commands are often found within the resources section of a job, so the first reset command that is encountered after an identified banner page is used to end the job.

EXAMPLE: With reference to the above diagram (Fig. 21.5), the following commands would be used to split the data at point A and create two jobs, both with a resources page, banner page and a page containing 'print' data:

```
STRING1=ALPHA
IDENTIFY1=1@5
ACTION1=INCLUDE,RELEASE,ENDJOB@RESET
```

The STRING1 command defines Identification String ALPHA, while IDENTIFY1 searches for it on line five of each page. ALPHA is not found on the first page (RESOURCES1) so this is released and the next page is checked

(BANNER1). ALPHA is found so the INCLUDE statement in the ACTION1 command holds the job until a PCL reset is encountered on page three (PRINT DATA1), at which point the data is split and the job released by the RELEASE statement. The process is repeated for the remaining pages within the data.

ENDJOB@NEXTBANNER

The ENDJOB@NEXTBANNER statement ends a job when the next banner page within the data is encountered.

ENDJOB@NEXTBANNER/RESET

The ENDJOB@NEXTBANNER/RESET statement ends a job either before the next banner page or at the next reset command found within the data.

Processing IBM Channel Data and CCWs

Banner Processing can be configured to recognise IBM Channel data and interpret Channel Control Words (CCW) to position the associated data on the correct line for processing. This is achieved by defining the PROCESS and CCW commands within the Banner Processing Parameter File.

NOTE: If the PASSCHANNELS parameter has been set to YES within the associated Task Configuration File, Banner Processing will not operate correctly. To overcome this, the PASSCHANNELS parameter should be set to CCW.

PROCESS

The PROCESS command determines whether to process IBM Channel data and CCWs.

NOTE: The PROCESS command should only appear once within the Parameter File.

Format

PROCESS has the following format:

```
PROCESS=xxxxxxxx
```

Where the value xxxxxxxx is one of the following:

- **ALL** Enable
- **CCW** Enable
- **CHTUNNEL** Enable
- **NORMAL** Disable

CCWxx

The CCWxx command determines what the default character associated with the value xx, in the CCW Action Translate Table, should be modified to.

Format

CCWxx has the following format:

```
CCWxx=hh
```

- **CCWxx** (hex) Determines the CCW value that is to be modified in the CCW Action Translate Table.
- **hh** (hex) Determines the value that the CCW defined by CCWxx, is to be changed to in the CCW Action Translate Table.

Binary Value

The binary value of the CCW requirement is shown in the following table.

NOTE: Bit seven is the High Order (HO) bit.

Binary bit	Value	Description
7	1	Ignore data associated with the CCW when searching for an Identification String.
6	1	After processing the data associated with the CCW, action as a reset.
5	1	After processing the data associated with the CCW, go to the logical top of form within Banner Processing.
4	1	Indicates a Channel throw. After processing the data associated with the CCW go to the calculated line number.
3 through 0	0 - F	The number of lines to logically move within Banner Processing, after processing any associated data.

EXAMPLE: The command:

CCW1B=83 (binary 10000011)

Would result in the action for CCW 1B becoming:

- Do not process any associated data within Banner Processing.
- Logically move down three lines within Banner Processing.

Where a CCW is specified as a Channel Throw, Banner Processing will make a 'best attempt' calculation as to which line it refers to by interrogating the Input Module line number. The calculation is not always accurate but as normal banner pages do not have any Channel Throws this does not normally cause a problem.

For further details, see Section 25, Channel Tunnelling, Section 28, Data Formats, and Appendix A, Translate Tables.

Dumping

For diagnostic purposes, Banner Processing can use the Dumping facility available to all TN Interfaces. It is enabled by setting the DUMPJOBSPLIT parameter.

NOTE: If the DUMP parameter is not enabled, the DUMPJOBSPLIT parameter is ignored.

For further details, see Section 24, Dumping.

Example Process

The following diagram (Fig. 21.6) provides an example of a typical IBM banner page, followed by sample Banner Processing Parameter Files that could be used to perform various actions.

Fig. 21.6
Example IBM banner page

```

BBBBBBBBBB OOOOOOOOOO BBBBBBBBBB DDDDDDDDD FFFFFFFF RRRRRRRRRR TTTTTTTTTT
BBBBBBBBBB OOOOOOOOOO BBBBBBBBBB DDDDDDDDD FFFFFFFF RRRRRRRRRR TTTTTTTTTT
BB BB OO OO BB BB DD DD FF RR RR TT
BB BB OO OO BB BB DD DD FF RR RR TT
BB BB OO OO BB BB DD DD FF RR RR TT
BBBBBBBBBB OO OO BBBBBBBBBB DD DD FFFFFFFF RRRRRRRRRR TT
BBBBBBBBBB OO OO BBBBBBBBBB DD DD FFFFFFFF RRRRRRRRRR TT
BB BB OO OO BB BB DD DD FF RR RR TT
BB BB OO OO BB BB DD DD FF RR RR TT
BB BB OO OO BB BB DD DD FF RR RR TT
BBBBBBBBBB OOOOOOOOOO BBBBBBBBBB DDDDDDDDD F F RR RR TT
BBBBBBBBBB OOOOOOOOOO BBBBBBBBBB DDDDDDDDD F F RR RR TT

```

```

JJJJJJJJJJ OOOOOOOOOO BBBBBBBBBB 11 6666666666 444 9999999999 7777777777
JJJJJJJJJJ OOOOOOOOOO BBBBBBBBBB 111 66666666666 4444 99999999999 7777777777
JJ OO OO BB BB 1111 66 66 44 44 99 99 77 77
JJ OO OO BB BB 11 66 66 44 44 99 99 77
JJ OO OO BB BB 11 66 66 44 44 99 99 77
JJ OO OO BBBBBBBBBB 11 66666666666 44444444444 99999999999 77
JJ OO OO BBBBBBBBBB 11 6666666666666 4444444444444 9999999999999 77
JJ OO OO BB BB 11 66 66 44 99 99 77
JJ JJ OO OO BB BB 11 66 66 44 99 99 77
JJ JJ OO OO BB BB 11 66 66 44 99 99 77
JJJJJJJJ OOOOOOOOOO BBBBBBBBBB 1111111111 6666666666666 44 9999999999999 77
JJJJJJ OOOOOOOOOO BBBBBBBBBB 1111111111 66666666666 44 9999999999 77

```

```

**START**START**START**START**START**START**START**START**START**START**
*
* JOBID: JOB16497
* JOB NAME: BOBDFRT
* USER ID: BOBD
* SYSOUT CLASS: U
* OUTPUT GROUP: 2.1.1
* TITLE:
*
* DESTINATION: LOCAL1
* NAME: TRANSMIT FILES
* ROOM:
* BUILDING:
* DEPARTMENT:
* ADDRESS:
*
*
* PRINT TIME: 18:10:43
* PRINT DATE: 31 MAR 1994
* PRINTER NAME: PRT4851
* SYSTEM: IP01
*
**START**START**START**START**START**START**START**START**START**START**

```

Splitting the Data and Defining a System Field

The following Parameter File can be used to split the data associated with the above banner page (Fig. 21.6) and define a System Field:

```

STRING1=**START*
STRING2=* DESTINATION
STRING3=?-17-??-8#1-?
IDENTIFY1=1@28-35&&2@36-45&3@52
ACTION1=BEFORE ,MULTI1 , PRINTER1

```

The first three lines are used to define the Identification Strings and determine the System Field and the variable data to copy to it.

IDENTIFY1 is used to search for the Identification String ****START*** (STRING1) on lines 28 to 35 (actual line = 31), *** DESTINATION** (STRING2) on lines 36 to 45 (actual line = 40) and variable data on line 52 (which must be present).

IDENTIFY1 ignores the first 17 characters of line 52, and assigns the following eight characters (PRT4851) to Temporary System Field #1.

Once IDENTIFY1 is satisfied, ACTION1 splits the data before the (single) banner page, associates the job with Printer Number 1 (PRINTER1) and copies the variable data (PRT4851) assigned to Temporary System Field #1 to System Field #1.

Deleting the Banner Page

The following Parameter File can be used to delete the above banner page (Fig. 21.6):

```
STRING1=**START*
STRING2=* DESTINATION
IDENTIFY1=1@28-35&2@36-45
ACTION1=INCLUDE , MULTI1 , DELETE
```

The first two lines define the Identification Strings.

IDENTIFY1 is used to search for the Identification String ****START*** (STRING1) on lines 28 to 35 (actual line = 31) and *** DESTINATION** on lines 36 to 45 (actual line = 40).

When both the Identification Strings are located, ACTION1 deletes the (single) banner page.

