CoolSpools

By Ariadne Software

PROGRAMMERS GUIDE

2020

Table of Contents

Introduction

This CoolSpools Programmer's Guide for Version 6 of CoolSpools provides detailed information required by programmers who wish to interface their applications into CoolSpools Version 6 or who intend to take advantage of advanced features such as the use of exit programs.

For information on running the CoolSpools commands, refer to the CoolSpools User's Guide for the relevant product option.

The CoolSpools Spool Conversion API

The CoolSpools Spool Conversion API invokes the CoolSpools functions that convert spooled files to stream files. As such it provides an alternative to calling one of the CoolSpools command interfaces (CVTSPLPDF, CVTSPLXL etc.).

Where you wish to integrate CoolSpools into your applications, the CoolSpools Spool Conversion API may provide a more convenient interface than running a command, especially if you need to interface into CoolSpools from code written in a language such as RPG, COBOL, C or Java, or if you need to specify complex parameters.

RPG Copybooks

A number of source members are provided in file CS_SRCFILE for use with ILE RPG. These can simplify the calling of the CoolSpools Spool Conversion API by providing data definitions.

The members are:

CS_CVTAPID

This source member contains the definition of constants and data structures required for defining API parameters.

CS_CVTAPIP

This source member contains the definition of the program prototype required for calling the API program.

These members should be included in your programs by means of the /COPY directive, e.g.:

```
* CoolSpools Spool Conversion API - Constants and structures
/COPY CS_SRCFILE,CS_CVTAPID

* CoolSpools Spool Conversion API - Program Prototypes
/COPY CS_SRCFILE,CS_CVTAPIP
```

CoolSpools Spool Conversion API (CS_CVTAPIR)

The CoolSpools Spool Conversion API (*PGM object ***CS_CVTAPIR***) allows access to CoolSpools functionality to convert a system i spooled file to one of several different file formats.

Required parameter group			
1	Spooled file name	Input	CHAR(10)
2	Qualified job name	Input	CHAR(26)
3	Spooled file number	Input	BINARY(4)
4	Length of stream file name	Input	BINARY(4)
5	Stream file name	Input	CHAR(*)
6	To Format	Input	CHAR(10)
7	Stream file option	Input	CHAR(10)
Omissible parameter group 1			
8	Error structure	I-O	CHAR(*)
Omissible parameter group 2			
9	Length of option structure list	Input	BINARY(4)
10	Option structure list	Input	CHAR(*)
Omissible parameter group 3			
11	Return structure	I-O	CHAR(*)
Omissible parameter group 3			
12	Format of return structure	Input	CHAR(8)

Required Parameter Group

Spooled file name

INPUT; CHAR(10)

The name of the spooled file to be converted.

Qualified job name

INPUT; CHAR(26)

The job that created the spooled file.

The qualified job name has three parts:

job name CHAR(10)

A specific job name, or one of the following special values:

*	The job that is running this program. The rest of the job name parameter must be blank.
*SBMJOB	The job that submitted the job that is running this program. The rest of the job name parameter must be blank.

user name CHAR(10)

A specific user profile name, or blanks when the job name is * or *SBMJOB.

job number *CHAR(6).*

A specific job number, or blanks when the job name is * or *SBMJOB.

Spooled file number

INPUT; BINARY(4)

The unique number of the spooled file. The valid range is 1 through 999999.

The following special values are supported for this parameter:

0	Only one spooled file from the job has the specified file name, so the number of the spooled file is not necessary.
-1	This uses the highest-numbered spooled file with the specified file name.

Length of stream file name

INPUT; BINARY(4)

The length of the stream file name specified on the next parameter.

Stream file name

INPUT; CHAR(*)

The name of the stream file to which the output is written. The length of this name should be specified on the previous parameter.

This name can be overridden at run time by using of the CS_STM01 parameter structure.

The following special values are supported for this parameter:

**FROMFILE*	The stream file name is generated from the name of the spooled file converted (FROMFILE parameter) and an extension appropriate to the TOFMT parameter (e.g. .pdf for *PDF). The file is saved in the current directory.
**FTP*	The file will be output to an FTP server. The details will be specified on the FTP parameter.
*EXITPGM	The stream file name will be supplied in the CS_STM01 structure by an exit program.

To format

INPUT; CHAR(10)

The format to which the stream file is to be converted.

**PDF*	PDF
**XLS*	Excel format.
**TEXT*	Text format
**CSV*	Delimited file format. The separator may not necessarily be a comma.
**TIFF*	TIFF format
**SAV*	Compressed stream file archive format. The spooled file can subsequently be restored from this stream file using the CVTSTMSPLF command.
**RTF*	Rich Text Format. This format is suitable for use with most word processing software.
**HTML*	Basic HTML format. This format reproduces fonts, highlighting, underlining etc. but does not necessarily reproduce the layout of the spooled file exactly.
**HTXT*	Text-oriented HTML. This format reproduces the layout of the spooled file better than *HTML but

does not reproduce fonts, highlighting, underlining etc.

**HTMLCSS* HTML including Cascading Style Sheet features. This format reproduces the appearance of the spooled file best but may not be supported by older versions of browser software.

Stream file option

INPUT; CHAR(*)

The way in which data is written to the stream file.

This name can be overridden at run time by using of the CS_STM01 parameter structure.

The following values are supported for this parameter:

**NONE* If the file exists, it will not be replaced and a conversion error will occur. If the file does not exist, it will be created.

**REPLACE* The file will be replaced if it exists and created if it does not.

**ADD* Data will be added to the file if it exists and created if it does not. This option is not supported for formats to which data cannot be appended, such as RTF.

**UNIQUE* CoolSpools will generate a unique name for the file in the specified directory by appending a suffix to the name given on the TOSTMF parameter

**EXITPGM* The stream file option will be supplied in the CS_STM01 structure by an exit program.

Optional Parameter Group 1

Error code

I/O; CHAR(*)

The structure in which to return error information. The format of the structure is defined under “Error structure” below.

If this parameter is omitted, diagnostic and escape messages are issued to the application.

Error structure

The error structure conforms to the format of the standard IBM API structure.

This structure is called CS_ERR01 in CoolSpools copybooks.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Structure size	The size of the error structure	BINARY(4)
4	0004	Length of error data	The total length of the error data available in this structure.	BINARY(4)
8	0008	Error message	The message identifier of the error that occurred.	CHAR(7)
15	000F	Reserved	Reserved	CHAR(1)
16	0010	Error data	The substitution data associated with the error message	CHAR(*)

Optional Parameter Group 2

Length of option structure list data

INPUT; BINARY(4)

The length of the option structure list provided on the following parameter.

Option structure list

IINPUT; CHAR(*)

A list of option structures.

See the section on the option structure list below for an explanation of the option structure list. This list should have been prepared using the option structure list APIs provided by CoolSpools.

If this parameter is omitted, default values are assumed for all options.

Optional Parameter Group 3

Return structure

I/O; CHAR(*)

A structure returned to the calling program containing various feedback information relating to the conversion.

See Format of return structure below for further details.

Optional Parameter Group 4

Format of return structure

I/O; CHAR(*)

Specifies the format of the structure returned to the calling program containing various feedback information relating to the conversion.

The default format if this parameter is omitted is CS_RTN01.

Structure CS_RTN01 –Return structure (format 1)

Name	Description
CS_RTN01	The CS_RTN01 structure returns feedback information to the caller of the API.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Structure size	The size of the variable provided by the caller for the return structure.	BINARY(4)
4	0004	Data available	The size of the return structure passed back to the caller.	BINARY(4)
8	0008	Status	The status of the conversion. 0 – OK. The conversion was successful. -1 – Error. The conversion ended in error. 1 – Warning. The conversion completed but one or more warning messages were issued.	BINARY(4)
12	000C	Number of stream files	The number of stream files created.	BINARY(4)
16	0010	Offset to stream file names	The offset from the start of this return structure to the list of the names of the stream files output.	BINARY(4)
20	0014	Length of stream file name list	The total length of the list of stream file names.	BINARY(4)

24	0018	Number of email messages	The number of email messages created.	BINARY(4)
28	001C	Offset to list of email ids	The offset from the start of this return structure to the list of email identifiers created. The list begins at the offset from the start of this structure specified at hex offset 10.	BINARY(4)
32	0020	Length of email id list	The total length of the list of email identifiers.	BINARY(4)
36	0024	Stream file names	A list of the names of the stream files output during the conversion. Each entry in the list consists of: a) The 2-byte length of the name b) The name itself Each name occurs immediately after the previous name with no padding.	CHAR(*)
*	*	Email ids	A list of the email identifiers created by CoolSpools Email during the conversion. The list begins at the offset from the start of this structure specified at hex offset 1C. Each entry is exactly 32 bytes long.	ARRAY OF CHAR(32)

Structure CS_RTN02 –Return structure (format 2)

Name	Description
CS_RTN02	The CS_RTN02 structure returns feedback information to the caller of the API. It is intended for use with spooled file conversion to spooled files (CVTSPLSPLF command or to-format *SPLF) and returns details of the spooled file(s) created to the caller.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Structure size	The size of the variable provided by the caller for the return structure.	BINARY(4)
4	0004	Data available	The size of the return structure passed back to the caller.	BINARY(4)
8	0008	Status	The status of the conversion. 0 – OK. The conversion was successful. -1 – Error. The conversion ended in error. 1 – Warning. The conversion completed but one or more warning messages were issued.	BINARY(4)
12	000C	Number of spooled files	The number of spooled files created.	BINARY(4)
16	0010	Offset to sspooled file names	The offset from the start of this return structure to the list of the names of the spooled files output.	BINARY(4)
20	0014	Length of spooled file name list	The total length of the list of spooled file names.	BINARY(4)
24	0018	Spooled file names	A list of the names of the spooled files output during the conversion. Each entry in the list is a structure in the format shown below.	CHAR(*)

Format of entries in the list of spooled file names

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Spooled file name	The name of the spooled file	CHAR(10)
10	000A	Spooled file job name	The name of the job in which the spooled file was created.	CHAR(10)
20	0014	Spooled file job user	The user profile of the job in which the spooled file was	CHAR(10)

			created.	
30	001E	Spooled file job number	The number of the job in which the spooled file was created.	CHAR(6)
36	0024	Spooled file number	The number of the spooled file	BINARY(4)

Structure CS_RTN03 –Return structure (format 3)

Name	Description
CS_RTN03	The CS_RTN03 structure returns feedback information to the caller of the API. It is similar to CS_RTN01 but includes details of the spooled file that was converted.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Structure size	The size of the variable provided by the caller for the return structure.	BINARY(4)
4	0004	Data available	The size of the return structure passed back to the caller.	BINARY(4)
8	0008	Spooled file name	The name of the spooled file converted	CHAR(10)
18	0012	Spooled file job name	The name of the job in which the spooled file was created.	CHAR(10)
28	001C	Spooled file job user	The user profile of the job in which the spooled file was created.	CHAR(10)
38	0026	Spooled file job number	The number of the job in which the spooled file was created.	CHAR(6)
44	002C	Spooled file number	The number of the spooled file	BINARY(4)
48	0030	Status	The status of the conversion. 0 – OK. The conversion was	BINARY(4)

			successful. -1 – Error. The conversion ended in error. 1 – Warning. The conversion completed but one or more warning messages were issued.	
52	0034	Number of stream files	The number of stream files created.	BINARY(4)
56	0038	Offset to stream file names	The offset from the start of this return structure to the list of the names of the stream files output.	BINARY(4)
60	003C	Length of stream file name list	The total length of the list of stream file names.	BINARY(4)
64	0040	Number of email messages	The number of email messages created.	BINARY(4)
68	0044	Offset to list of email ids	The offset from the start of this return structure to the list of email identifiers created. The list begins at the offset from the start of this structure specified at hex offset 10.	BINARY(4)
72	0048	Length of email id list	The total length of the list of email identifiers.	BINARY(4)
76	004C	Stream file names	A list of the names of the stream files output during the conversion. Each entry in the list consists of: a) The 2-byte length of the name b) The name itself Each name occurs immediately after the previous name with no padding.	CHAR(*)
*	*	Email ids	A list of the email identifiers created by CoolSpools Email during the conversion. The list begins at the offset from the start of this structure specified	ARRAY OF CHAR(32)

			at hex offset 1C. Each entry is exactly 32 bytes long.	

Structure CS_RTN04 –Return structure (format 4)

Name	Description
CS_RTN04	The CS_RTN04 structure returns feedback information to the caller of the API. It is intended for use with spooled file conversion to spooled files (CVTSPLSPLF command or to-format *SPLF) and returns details of the spooled file(s) created to the caller. It is similar to CS_RTN04 but also includes details of the spooled file that was converted.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Structure size	The size of the variable provided by the caller for the return structure.	BINARY(4)
4	0004	Data available	The size of the return structure passed back to the caller.	BINARY(4)
8	0008	Spooled file name	The name of the spooled file converted	CHAR(10)
18	0012	Spooled file job name	The name of the job in which the spooled file was created.	CHAR(10)
28	001C	Spooled file job user	The user profile of the job in which the spooled file was created.	CHAR(10)
38	0026	Spooled file job number	The number of the job in which the spooled file was created.	CHAR(6)
44	002C	Spooled file number	The number of the spooled file	BINARY(4)
48	0030	Status	The status of the conversion. 0 – OK. The conversion was successful.	BINARY(4)

			-1 – Error. The conversion ended in error. 1 – Warning. The conversion completed but one or more warning messages were issued.	
52	0034	Number of spooled files	The number of spooled files created.	BINARY(4)
56	0038	Offset to spooled file names	The offset from the start of this return structure to the list of the names of the spooled files output.	BINARY(4)
60	003C	Length of spooled file name list	The total length of the list of spooled file names.	BINARY(4)
64	0040	Spooled file names	A list of the names of the spooled files output during the conversion. Each entry in the list is a structure in the format shown below.	CHAR(*)

Format of entries in the list of spooled file names

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Spooled file name	The name of the spooled file	CHAR(10)
10	000A	Spooled file job name	The name of the job in which the spooled file was created.	CHAR(10)
20	0014	Spooled file job user	The user profile of the job in which the spooled file was created.	CHAR(10)
30	001E	Spooled file job number	The number of the job in which the spooled file was created.	CHAR(6)
36	0024	Spooled file number	The number of the spooled file	BINARY(4)

Option structure list

CoolSpools provides a large number of user-definable options which determine how spooled files are processed. To provide API program parameters for each possible option would have resulted in an enormously complex API parameter list. CoolSpools therefore allows you to specify all of the option data through a single API parameter, but the option data must be organized in a specific way, namely in the form of a CoolSpools *option structure list*.

CoolSpools provides a set of APIs for managing the option structure list. You are strongly recommended to use these APIs for creating and processing option structure lists and their contents.

The option structure list consists of one or more option structures. Each option structure comprises a generic header area and a data portion the format of which is specific to the particular option structure, for example:

Option structure 1 Header
Option structure 1 Data
Option structure 2 Header
Option structure 2 Data
…
Option structure n Header
Option structure n Data

The following concepts should be considered when managing the option structure list.

1. Each option structure defines a set of options that control the way in which CoolSpools operates.
2. Each option structure is identified by a format name such as CS_PDF01 which indicates the type of structure that is being defined.
3. Option structures are not normally mandatory (i.e. the minimum number of occurrences of the structure is zero) but in some circumstances a structure may become mandatory. For example, if the stream file name is passed to the CoolSpools Spool Conversion API as *EXITPGM, indicating that the actual name will be provided by an exit program, you must provide a CS_STM01 structure defining the name to be used before CoolSpools tries to open the stream file
4. Each type of option structure has a maximum number of occurrences which is supported by CoolSpools. For some structures the maximum is 1; for others, many structures are permitted (typically up to 100).
5. Some structures are only permitted in conjunction with certain values of the to-format parameter. For example, the CS_PDF01 structure which defines PDF-related options can only be specified with to-format *PDF.

6. Certain structures are *overridable*, in other words their initial value may be modified at run-time, whereas other structures are *non-overridable* and can only be defined at startup, before the CoolSpools Spool Conversion API is called. For example, the CS_SPT01 structure which defines splitting options is non-overridable and can only be added to the list at startup since splitting decisions are taken before exit programs are called. In contrast, the CS_EMT01 structure which defines email recipients can be added to the list at any point other than the special *PAGECTL exit point.
7. Certain structures may only be added from particular exit points. For example, the CS_STM01 structure which defines the stream file name cannot be added from a page-level exit point (*PAGESTR or *PAGEEND) because by the time that exit point is called, the stream file has already been opened and it does not make sense to try to change the name of a stream file half way through creating it.
8. The special CS_FBK01 structure, which allows pages to be excluded from processing, can only be called from the special *PAGECTL exit point and is the only structure that can be added to the list at that point.
9. All of these various attributes and restrictions related to structures are documented in the tables below.

Generic option header structure

The option header structure must take the following format, which is also defined in RPG copybook AR_APIFNCD.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Option id	A unique option identifier for the option structure	BINARY(4)
4	0004	Format	The format of the data that follows. This must be a recognized CoolSpools option data format name (see list below).	CHAR(8)
12	000C	Priority	The priority of the option structure being defined. The priority determines, along with other factors, whether a particular option structure will be in effect. This must be one of the following values: *1* – Primary scope. So long as the page scope is appropriate, the option structure will always be selected and will influence CoolSpools processing. *2* – Secondary. The option structure will only be selected and influence CoolSpools processing if the page scope is appropriate and no other structure of this format with primary priority has already been selected. *3* – Tertiary. This is reserved for system use and applies to system-generated default structures.	CHAR(1)
13	000D	Status	The status of the option structure. *A* – Active. The structure is still available to influence processing. *D* – Deleted. The structure is no longer available to influence processing.	CHAR(1)

14	000E	Data length	Length of the data part of this option structure (excludes the length of this header).	BINARY(4)
18	0012	Scope from-page	The first page to which the option structure relates. The options controlled by the structure being defined will only be in effect for pages that are greater than or equal to the value specified here.	BINARY(4)
22	0016	Scope to-page	The last page to which the option structure relates. The options controlled by the structure being defined will only be in effect for pages that are less than or equal to the value specified here. The value specified here must be greater than or equal to the value specified for the from-page.	BINARY(4)
26	001A	Context	The stage of processing at which the structure was added, **STRUP* Startup. The structure was added before the CoolSpools Spool Conversion API was called. **SYSDFT* System default. The structure was system-generated because no default value had been provided. *SPLFSTR The structure was added by an exit program called at the *SPLFSTR exit point. *STMFSTR The structure was added by an exit program called at the *STMFSTR exit point. *PAGESTR The structure was added by an exit program called at the *PAGESTR exit point. *SPLFEND The structure was added by an exit program called at the *SPLFEND exit point.	CHAR(10)

			*STMFEND The structure was added by an exit program called at the *STMFEND exit point. *PAGEEND The structure was added by an exit program called at the *PAGEEND exit point. *PAGECTL The structure was added by an exit program called at the *PAGECTL exit point.	
36	0024	Reserved	Reserved. Must be set to binary zeros.	CHAR(18)

The header must be followed immediately by the option data. The option data must conform to the appropriate structure listed below, corresponding to the format defined in the header.

These structures are also defined in copybook CS_CVTAPID.

Each structure is defined as ending with a one-byte reserved area required for internal processing.

For each structure documented below, the table indicates the maximum number of structures and the exit points at which the structure may be added to the option structure list. All structures may be added to the option structure list at startup other than the special feedback structure CS_FBK01, which may be added only from the *PAGECTL exit point.

Structure CS_BMK01 – Bookmarks by key options (format 1)

Name	Description
CS_BMK01	The CS_BMK01 structure defines options for selecting bookmark text from a spooled file by key string.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

At least one CS_BMK01 or CS_BMK02 structure is required if keyed bookmarks are indicated (bookmark option in CS_PDF01 is *KEY or *POSKEY.

Valid only with output format *PDF.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of the key string	The length of the data in the following field.	BINARY(2)
2	0002	Key string	The key string which selects bookmark data.	CHAR(512)
514	0202	Occurrence	The occurrence on the page of this key string which selects bookmarks text. The valid range is 1-999.	BINARY(4)
518	0206	Reserved	Reserved.	CHAR(4)
522	020A	Horizontal offset	The offset from the start of the key string to start of the data to be selected as a bookmark.	DEC(7,3)
526	020E	Length	The length of the data to be selected as a bookmark.	DEC(7,3)
530	0212	Unit	The units in which the coordinates and length are defined. ***ROWCOL*** – Rows and columns This is now the only supported option.	CHAR(10)

Structure CS_BMK02 – Bookmarks by key options (format 2)

Name	Description
CS_BMK02	The CS_BMK02 structure defines options for selecting bookmark text from a spooled file by key string. Unlike CS_BMK01, all coordinates must be defined in terms of rows and columns, the exact location on the page where the key string should be checked for can be specified, and a vertical offset from the key string to the bookmark string can also be given.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

At least one CS_BMK01 or CS_BMK02 structure is required if keyed bookmarks are indicated (bookmark option in CS_PDF01 is *KEY or *POSKEY.

Valid only with output format *PDF.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of the key string	The length of the data in the following field.	BINARY(2)
2	0002	Key string	The key string which selects bookmark data.	CHAR(512)
514	0202	Line number where key should be checked for	The line number on the page where the key string is to be checked for. ***-1*** – Any line. All lines are checked for the key string.	BINARY(4)
518	0206	Column number where key should be checked for	The column number in the line where the key string is to be checked for. ***-1*** – Any column. All columns are checked for the key string.	BINARY(4)
522	020A	Occurrence	The occurrence on the page of this key string which selects bookmarks text. The valid range is 1-999.	BINARY(4)
526	020E	Vertical offset	The vertical offset from the start of	BINARY(4)

			the key string to start of the data to be selected as a bookmark, in rows/lines	
530	0212	Horizontal offset	The horizontal offset from the start of the key string to start of the data to be selected as a bookmark, in columns/characters.	BINARY(4)
534	0216	Length	The length of the data to be selected as a bookmark, in columns/characters.	BINARY(4)

Structure CS_BMP01 – Bookmarks by position options

Name	Description
CS_BMP01	The CS_BMP01 structure defines options for selecting bookmark text from a spooled file on a positional basis.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

At least one CS_BMP01 structure is required if positional bookmarks are indicated (bookmark option in CS_PDF01 is *POS or *POSKEY.

Valid only with output format *PDF.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Vertical coordinate or line number	The vertical coordinate or line number of the data to be selected as a bookmark.	DEC(7,3)
4	0004	Horizontal coordinate or column	The horizontal coordinate or column of the start of the data to be selected as a bookmark.	DEC(7,3)
8	0008	Length	The length of the data to be selected as a bookmark.	DEC(7,3)
12	000C	Unit	The units in which the coordinates and length are defined. ***ROWCOL*** – Rows and columns This is now the only supported option.	CHAR(10)

Structure CS_CFI01 – Instructions for converting font identifiers

Name	Description
CS_CFI01	The CS_CFI01 structure provides a means of overriding CoolSpools font processing to define explicitly how a given font should be implemented, where that font is specified in the spooled file by means of a font identifier (e.g. with the DDS FONT keyword).

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Font identifier	The font identifier to which this instruction relates, e.g. 11 = Courier 10 CPI.	BINARY(4)
4	0004	Font size	The font size to which this instruction relates. This will be interpreted as a pitch or point size depending on the nature of the font identifier. Specify a value between 4 and 36 or the following special value: ***0*** – The font size is implied by the font identifier.	BINARY(4)
8	0008	Implement using font	How to implement the specified font. This can be: i) the IFS path of a Postscript font held in the IFS (e.g. **qibm/ProdData/OS400/Fonts/PSFonts/Latin/COU.PFB**) ii) the IFS path of a font resource object (font character set or coded font), e.g. **/qsys.lib/qfntcpl.lib/X0AOA.FNT**	CHAR(256)

			RSC) iii) one of the special values: ***COURIER** – Courier ***COURIERB** – Courier Bold ***COURIERO** – Courier Oblique ***COURIERBO** – Courier Bold Oblique ***HELVETICA** – Helvetica (in practice the font used may be Arial) ***HELVB** – Helvetica Bold ***HELVO** – Helvetica Oblique ***HELVBO** – Helvetica Bold Oblique ***TIMES** – Times Roman ***TIMESB** – Times Roman Bold ***TIMESI** – Times Roman Italic ***TIMESBI** – Times Roman Bold Italic ***SYMBOL** - Symbol ***DINGBATS** – Zapf Dingbats	
264	0108	Implement using font size	The font size to be used with the font specified in the previous field, ***0*** – The font size implied by the font identifier or specified in the Font Size field above.	BINARY(4)

Structure CS_CFR01 – Instructions for converting font resources

Name	Description
CS_CFR01	The CS_CFR01 structure provides a means of overriding CoolSpools font processing to define explicitly how a given font should be implemented, where that font is specified in the spooled file by means of a font resource name (e.g. with the DDS FNTCHRSET or CDEFNT keywords).

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Font resource name	The font resource name to which this instruction relates. This should be either a font character set or a code font name.	CHAR(8)
4	0004	Font size	The font size to which this instruction relates. This will be interpreted as a pitch or point size depending on the nature of the font resource. Specify a value between 4 and 36 or the following special value: ***0*** – The font size is implied by the font resource name.	BINARY(4)
12	000C	Implement using font	How to implement the specified font. This can be: i) the IFS path of a Postscript font held in the IFS (e.g. **qibm/ProdData/OS400/Fonts/PSFonts/Latin/COU.PFB**) ii) the IFS path of a font resource object (font character set or coded font), e.g. **/qsys.lib/qfntcpl.lib/X0AOA.FNT**	CHAR(256)

			RSC) iii) one of the special values: ***COURIER** – Courier ***COURIERB** – Courier Bold ***COURIERO** – Courier Oblique ***COURIERBO** – Courier Bold Oblique ***HELVETICA** – Helvetica (in practice the font used may be Arial) ***HELVB** – Helvetica Bold ***HELVO** – Helvetica Oblique ***HELVBO** – Helvetica Bold Oblique ***TIMES** – Times Roman ***TIMESB** – Times Roman Bold ***TIMESI** – Times Roman Italic ***TIMESBI** – Times Roman Bold Italic ***SYMBOL** - Symbol ***DINGBATS** – Zapf Dingbats	
268	010C	Implement using font size	The font size to be used with the font specified in the previous field, ***0*** – The font size implied by the font resource or specified in the Font Size field above.	BINARY(4)

Structure CS_CLR01 – Color options

Name	Description
CS_CLR01	Defines the colors to be used.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	Yes	Yes	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Text color	The color to be used for text in the document. Either specify a predefined color name from the following list or an RGB color number An RGB (Red-Green-Blue) color number is similar to a color number that can be used in HTML. It is a string consisting of six hexadecimal digits (0-F). The first two digits represent the red color value (00-FF), The next two digits represent the green color value (00-FF), The last two digits represent the blue color value (00-FF), For example, white is FFFFFF, while black is 000000, and red is FF0000. Do not prefix the color number with a hash sign (#) as you would in HTML. Predefined color names that can be used are ***ALICEBLUE** ***ANTIQUEWHITE** ***AQUA** ***AQUAMARINE** ***AZURE**	CHAR(20)

			***BEIGE** ***BISQUE** ***BLACK** ***BLANCHEDALMOND** ***BLUE** ***BLUEVIOLET** ***BROWN** ***BURLYWOOD** ***CADETBLUE** ***CHARTREUSE** ***CHOCOLATE** ***CORAL** ***CORNFLOWERBLUE** ***CORNSILK** ***CRIMSON** ***CYAN** ***DARKBLUE** ***DARKCYAN** ***DARKGOLDENROD** ***DARKGRAY** ***DARKGREY** ***DARKGREEN** ***DARKKHAKI** ***DARKMAGENTA** ***DARKOLIVEGREEN** ***DARKORANGE** ***DARKORCHID** ***DARKRED** ***DARKSALMON** ***DARKSEAGREEN** ***DARKSLATEBLUE** ***DARKSLATEGRAY** ***DARKSLATEGREY** ***DARKTURQUOISE** ***DARKVIOLET** ***DEEPPINK** ***DEEPSKYBLUE** ***DIMGRAY** ***DIMGREY** ***DODGERBLUE** ***FELDSPAR** ***FIREBRICK** ***FLORALWHITE** ***FORESTGREEN** ***FUCHSIA** ***GAINSBORO** ***GHOSTWHITE** ***GOLD** ***GOLDENROD** ***GRAY**	

			***GREY** ***GREEN** ***GREENYELLOW** ***HONEYDEW** ***HOTPINK** ***INDIANRED** ***INDIGO** ***IVORY** ***KHAKI** ***LAVENDER** ***LAVENDERBLUSH** ***LAWNGREEN** ***LEMONCHIFFON** ***LIGHTBLUE** ***LIGHTCORAL** ***LIGHTCYAN** ***LIGHTGOLDENROD** ***LIGHTGRAY** ***LIGHTGREY** ***LIGHTGREEN** ***LIGHTPINK** ***LIGHTSALMON** ***LIGHTSEAGREEN** ***LIGHTSKYBLUE** ***LIGHTSLATEBLUE** ***LIGHTSLATEGRAY** ***LIGHTSLATEGREY** ***LIGHTSTEELBLUE** ***LIGHTYELLOW** ***LIME** ***LIMEGREEN** ***LINEN** ***MAGENTA** ***MAROON** ***MEDIUMAQUAMARINE** ***MEDIUMBLUE** ***MEDIUMORCHID** ***MEDIUMPURPLE** ***MEDIUMSEAGREEN** ***MEDIUMSLATEBLUE** ***MEDIUMSPRINGGREEN** ***MEDIUMTURQUOISE** ***MEDIUMVIOLETRED** ***MIDNIGHTBLUE** ***MINTCREAM** ***MISTYROSE** ***MOCCASIN** ***NAVAJOWHITE** ***NAVY** ***OLDLACE**	

			***OLIVE** ***OLIVEDRAB** ***ORANGE** ***ORANGERED** ***ORCHID** ***PALEBLUE** ***PALEBROWN** ***PALECYAN** ***PALEGOLDENROD** ***PALEGRAY** ***PALEGREY** ***PALEGREEN** ***PALEMAG** ***PALETURQUOISE** ***PALEVIOLETRED** ***PALEYLW** ***PAPAYAWHIP** ***PEACHPUFF** ***PERU** ***PINK** ***PLUM** ***POWDERBLUE** ***PURPLE** ***RED** ***ROSYBROWN** ***ROYALBLUE** ***SADDLEBROWN** ***SALMON** ***SANDYBROWN** ***SEAGREEN** ***SEASHELL** ***SIENNA** ***SILVER** ***SKYBLUE** ***SLATEBLUE** ***SLATEGRAY** ***SLATEGREY** ***SNOW** ***SPRINGGREEN** ***STEELBLUE** ***TAN** ***TEAL** ***THISTLE** ***TOMATO** ***TURQUOISE** ***VIOLET** ***VIOLETRED** ***WHEAT** ***WHITE** ***WHITESMOKE**	

			***YELLOW** ***YELLOWGREEN**	
10	000A	Background color	The color to be used for the background on which text is presented. Options are the same as for the text color above.	CHAR(20)

Structure CS_CST01 – Custom Page Size

Name	Description
CS_CST01	Custom page size. Specifies the dimensions of a custom page size. Required if the paper size defined in CS_PGS01 is *CUSTOM.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Page Width	The width of the page.	DEC(7,3)
4	0004	Page Length	The length of the page.	DEC(7,3)
8	0008	Unit	The unit in which these dimensions are specified: **INCH* – inches **MM* – millimeters **CM* - centimeters	CHAR(10)

Structure CS_CSV01 – CSV options

Name	Description
CS_CSV01	The CS_CSV01 structure options for use with output format *CSV.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	Yes	Yes	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Record delimiter	How the end of records is denoted in the output file. Options are: ***CRLF** – A carriage return and linefeed is inserted at the end of each record. ***CR** – A carriage return alone is inserted at the end of each record. ***LF** – A linefeed alone is inserted at the end of each record.	CHAR(10)
10	000A	String delimiter	The character to be used to enclose strings (text) in the output file, e.g. double quotes. A blank indicates that there will be no string delimiter.	CHAR(1)
11	000B	Field delimiter	The character to be used to separate fields in the output file, e.g. a comma. Specify x'05' for tabs.	CHAR(1)
12	000C	Column separator	The character in the spooled file which will trigger the creation of a new column (field) in the output file. The following special value may be used.	CHAR(1)

			x'00' – Columns will not be created based on a column separator but CoolSpools will rely instead on the internal structure of the spooled file data.	
13	000D	Number of column separators	The number of consecutive characters of the type defined in the previous field which will trigger the creation of a new column (field) in the output file. If the previous field is x'00', this field must be set to zero. If the previous field is not x'00', this field must not be set to zero.	BINARY(4)
17	0011	Column allocation method	The method used by CoolSpools to allocate text in the spooled file to columns in the output. Options are: ****OLD*** – The method used by CoolSpools in releases prior to Version 6 is used. This either method relies on the organization of the data in the spooled file or splits the data in the spooled file up into tokens based on the column separators defined on this parameter. ****NEW*** – This method uses statistical techniques to identify patterns in the spooled file data and uses the information obtained in this way to determine which rows in the report are data and which headings and where columns of text and numbers start and end.	CHAR(10)
27	001B	Page headings	How to handle page headings. ****FIRST*** – Keep the first occurrence of a page heading but drop all others. ****ALL*** – Keep all occurrences of a page heading. ****NONE*** – Drop all occurrences of	CHAR(10)

			a page heading.	
37	0025	Column headings	How to handle column headings. ***FIRST** – Keep the first occurrence of a column heading but drop all others. ***ALL** – Keep all occurrences of a column heading. ***NONE** – Drop all occurrences of a column heading.	CHAR(10)
47	002F	Spooled file currency symbol	The currency symbol used in the spooled file. This enables CoolSpools to locate and properly convert currency data in the spooled file. Specify the currency symbol used or the following special value: ***x'01'** – The currency symbol defined by the QCURSYM system value	CHAR(1)
48	0030	Spooled file decimal point	The decimal point used in the spooled file. This enables CoolSpools to locate and properly convert decimal data in the spooled file. Specify the decimal point symbol used or one of the following special values: ***x'01'** – The decimal point symbol defined by the QDECFMT system value ***x'02'** – The decimal point symbol defined by the DECFMT job attribute.	CHAR(1)
49	0031	Spooled file thousands separator	The thousands separator used in the spooled file. This enables CoolSpools to locate and properly convert decimal data in the spooled file. Specify the thousands separator used or one of the following special values: ***x'01'** – The thousands separator	CHAR(1)

			defined by the QDECFMT system value ***x'02'*** – The thousands separator defined by the DECFMT job attribute.	
50	0032	Spooled file date format	The date format used in the spooled file. This enables CoolSpools to locate and properly convert dates in the spooled file. Specify one of the following special values: ****SYSVAL*** – The date format defined by the QDATFMT system value ****JOB*** – The date format defined by the DATFMT job attribute. ****DMY*** – Day-Month-Year format ****MDY*** – Month-Day-Year format ****YMD*** – Year- Month-Day format	CHAR(10)
60	003C	Spooled file date separator	The date separator used in the spooled file. This enables CoolSpools to locate and properly convert dates in the spooled file. Specify the date separator character used or one of the following special values: ***x'01'*** – The date separator defined by the QDATSEP system value ***x'02'*** – The date separator defined by the DATSEP job attribute.	CHAR(1)
61	003D	Word for "page"	The word for "Page" used in the spooled file. This helps CoolSpools locate page numbers in the spooled file and identify page headings and footings. Specify the word for "Page" used in the spooled file or the following special values	CHAR(20)

			***DFT** – CoolSpools will retrieve the word for “Page” from the text of message id CVT0008 in message file CP_MSGF.	

Structure CS_DBC01 – DBCS options

Name	Description
CS_DBC01	The CS_DBC01 structure defines various DBCS options.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	DBCS Coded font	The coded font to be used to implement DBCS fonts in the spooled file. CoolSpools uses this information where the font specified in the spooled file attributes is unavailable. You may use the following special value: ***SPLF** – Use the DBCS font specified on the IGCCDEFNT attribute of the spooled file.	CHAR(8)
8	0008	DBCS Coded font library	The library in which the coded font is located. You may specify one of the special values: ***LIBL** – Use the current library list to locate the font. ***CURLIB** – The font is located in the current library.	CHAR(10)
18	0012	DBCS font size	The size of the font to use for DBCS text. Specify a font size or use the following special value. ***-1** – Derive the appropriate font size from the attributes of the spooled file and the font information it contains.	DEC(5,1)

21	0015	DBCS data in Non-DBCS capable spooled file	The action to take when what appears to be DBCS data is found in a spooled file the attributes of which indicate that it is not DBCS-capable. Options are: ***NO*** – Treat the data that appears to be DBCS as SBCS ***YES*** – Treat the data that appears to be DBCS as DBCS, irrespective of the spooled file attribute.	CHAR(10)

Structure CS_DEV01 – Printer device

Name	Description
CS_DEV01	Print device. Allows the definition of the printer device for which the spooled file was designed. Where *DEVD is specified for the paper size (CS_PGS01 structure), this enables CoolSpools to derive the page size from the device description.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Device name	The name of the device on which this spooled file would normally be printed. CoolSpools can use this information to retrieve certain attributes from the printer device description.	CHAR(10)

Structure CS_EML01 – Email options

Name	Description
CS_EML01	The CS_EML01 structure defines email options. **N.B.** **This structure is still supported, and can be used to define email options when calling the CoolSpools Spool Converter API.** **However, CoolSpools itself now uses the enhanced CS_EML02 structure. Where CoolSpools commands are being run, exit programs should therefore reference the CS_EML02 structure to modify email options at run time.**

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	Yes	Yes	Yes	Yes	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Email option	Whether to send the output file by email. Options are: ***NO** – Do not email the output ***YES** – Email the output. Every time a new output file is created, it will be emailed as a single attachment to the recipients specified. ***ONE** – Email a single message. All output files created will be emailed together as attachments on a single message at the end of the conversion run.	CHAR(10)
10	000A	Delete after emailing	Whether the output file should be deleted after it has been emailed. This field must be blank if the email option is *NO. If the email option is *YES, possible values are:	CHAR(10)

			YES – Delete the output after it has been emailed. ***NO*** – Do not delete the output after it has been emailed.	
20	0014	Length of subject	The length of the data in the following field. This must be zero if the email option is *NO.	BINARY(2)
22	0016	Subject	The subject text of the email.	CHAR(50)
72	0048	Method	How the output is added to the email. This field must be blank if the email option is *NO. If the email option is *YES, possible values are: ***ATTACH*** – The file appears as an attachment. ***EMBED*** – The file is included in the text of the email. This is only possible with text-oriented output formats (*TEXT, *HTML, *HTXT, *HTMLCSS, *CSV).	CHAR(10)
82	0052	Priority	The priority assigned to the email. This field must be blank if the email option is *NO. If the email option is *YES, possible values are: ***N*** – Normal ***H*** – High ***L*** - Low	CHAR(1)
83	0053	Request confirmation of receipt	Whether the recipient is requested to confirm receipt. This field must be blank if the email option is *NO. If the email option is *YES, possible values are: ***Y*** – Confirmation of receipt is requested. ***N*** – Confirmation of receipt is not requested.	CHAR(1)

84	0054	Send multiple messages	When sending to multiple recipients, determines whether a separate message is sent to each recipient or one email is sent with multiple recipients. This field must be blank if the email option is *NO. If the email option is *YES, possible values are: ***Y*** – One email per recipient is sent. The person reading the email will not know who else it was sent to. ***N*** – One email is sent to a list of recipients. Each person reading the email will know who else it was sent to.	CHAR(1)
85	0055	Length of sender's email address	The length of the following field. This must be zero if the email option is *NO.	BINARY(2)
87	0056	Sender's email address	The email address of the sender. The following special value may be used: ****CURRENT*** – The details are retrieved from the system distribution directory for the user running the API.	CHAR(128)
215	00D7	Length of sender's name	The length of the following field. This must be zero if the email option is *NO.	BINARY(2)
217	00D9	Sender's name	The name of the sender. The following special value may be used: ****CURRENT*** – The details are retrieved from the system distribution directory for the user running the API.	CHAR(128)
345	0159	Message format	The format in which the message is sent. This field must be blank if the	CHAR(10)

			email option is *NO. If the email option is *YES, possible values are: ***BOTH** – Alternate HTML and plain text versions are sent. ***HTML** – Just an HTML version is sent. ***TEXT**– Just a plain text version is sent.	
355	0163	Length of message text	The length of the following field. This must be zero if the email option is *NO.	BINARY(2)
352	0164	Message text	The text of the message to be sent.	CHAR(*)

Structure CS_EML02 – Email options

Name	Description
CS_EML02	The CS_EML02 structure defines email options. It is an enhanced version of the CS_EML01 structure. Compared with CS_EML01, it allows for longer values of the subject field and includes some additional fields. **N.B.** **CS_EML02 is still supported, and can be used to define email options when calling the CoolSpools Spool Converter API.** **However, CoolSpools itself now uses the enhanced CS_EML02 structure. Where CoolSpools commands are being run, exit programs should therefore reference the CS_EML02 structure to modify email options at run time.**

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	Yes	Yes	Yes	Yes	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Email option	Whether to send the output file by email. Options are: ***NO** – Do not email the output ***YES** – Email the output. Every time a new output file is created, it will be emailed as a single attachment to the recipients specified. ***ONE** – Email a single message. All output files created will be emailed together as attachments on a single message at the end of the conversion run.	CHAR(10)
10	000A	Delete after emailing	Whether the output file should be deleted after it has been emailed. This field must be blank if the email option is *NO. If the email	CHAR(10)

			option is *YES, possible values are: ***YES** – Delete the output after it has been emailed. ***NO** – Do not delete the output after it has been emailed.	
20	0014	Length of subject	The length of the data in the following field. This must be zero if the email option is *NO.	BINARY(2)
22	0016	Subject	The subject text of the email.	CHAR(256)
278	0116	Method	How the output is added to the email. This field must be blank if the email option is *NO. If the email option is *YES, possible values are: ***ATTACH** – The file appears as an attachment. ***EMBED** – The file is included in the text of the email. This is only possible with text-oriented output formats (*TEXT, *HTML, *HTXT, *HTMLCSS, *CSV).	CHAR(10)
288	0120	Priority	The priority assigned to the email. This field must be blank if the email option is *NO. If the email option is *YES, possible values are: ***N** – Normal ***H** – High ***L** - Low	CHAR(1)
289	0121	Request confirmation of receipt	Whether the recipient is requested to confirm receipt. This field must be blank if the email option is *NO. If the email option is *YES, possible values are: ***Y** – Confirmation of receipt is requested. ***N** – Confirmation of receipt is not	CHAR(1)

			requested.	
290	0122	Send multiple messages	When sending to multiple recipients, determines whether a separate message is sent to each recipient or one email is sent with multiple recipients. This field must be blank if the email option is *NO. If the email option is *YES, possible values are: ***Y*** – One email per recipient is sent. The person reading the email will not know who else it was sent to. ***N*** – One email is sent to a list of recipients. Each person reading the email will know who else it was sent to.	CHAR(1)
291	0123	Length of sender's email address	The length of the following field. This must be zero if the email option is *NO.	BINARY(2)
293	0125	Sender's email address	The email address of the sender. The following special value may be used: ****CURRENT*** – The details are retrieved from the system distribution directory for the user running the API.	CHAR(128)
421	01A5	Length of sender's name	The length of the following field. This must be zero if the email option is *NO.	BINARY(2)
423	01A7	Sender's name	The name of the sender. The following special value may be used: ****CURRENT*** – The details are retrieved from the system distribution directory for the user running the API.	CHAR(128)
551	0227	Message format	The format in which the message is sent.	CHAR(10)

			This field must be blank if the email option is *NO. If the email option is *YES, possible values are: ***BOTH** – Alternate HTML and plain text versions are sent. ***HTML** – Just an HTML version is sent. ***TEXT**– Just a plain text version is sent.	
561	0163	Length of attachment name	The length of the following field. This must be zero if the email option is *NO.	BINARY(2)
563	0165	Attachment name	The name of the attachment, as it will appear in the received email message. There is a special value: ***TOSTMF** – The attachment name will be the name of the file being attached.	CHAR(64)
627	0273	Length of message text	The length of the following field. This must be zero if the email option is *NO.	BINARY(2)
629	0275	Message text	The text of the message to be sent.	CHAR(16384)
17013	4275	Message type	Determines how the contents of the preceding field are interpreted. Options are: ***MSG** – The message text field is interpreted as specifying the actual text of the message to send. ***STMF** - The message text field is interpreted as specifying the actual text of the message path of a stream file which contains the actual text of the message to send.	CHAR(10)
17023	427F	Zip attachment	Whether the attachment should be sent inside a zip file. Options are:	CHAR(1)

			Y – Yes, send the attachment(s) inside a zip file. ***N*** – No, do not send the attachment(s) inside a zip file.	
17024	4280	Length of zip file password	The length of the following field. This must be zero if no zip password is required.	BINARY(2)
17026	4282	Zip file password	The password to apply to the zip file, if any. There is a special value: ****NONE*** – No password is applied.	CHAR(32)
17058	42A2	Encrypted password supplied	Whether the password specified in the previous field is in the encrypted form returned by DSPENCPWD or not.	CHAR(1)
17059	42A3	Save email	Whether the email message should be saved or not. Saving the email allows it to be re-sent more easily. Options are: ***Y*** – Save the email ***N*** – No, do not save the email.	CHAR(1)
17060	42A4	Save for how many days	The retention period in days to assign to the saved email. Saved emails are deleted by running the DLTCMNMSG command. If DLTSAVMSG(*MSG) is specified, saved emails will be deleted after the number of days specified here.	BINARY(2)

Structure CS_EMT01 – Email-to options

Name	Description
CS_EMT01	The CS_EMT01 structure defines the recipients to whom output files are emailed.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	Yes	Yes	Yes	Yes	Yes	Yes	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of the recipient's email address	The length of the data in the following field.	BINARY(2)
2	0002	Recipient's email address	The email address to which the message should be sent. The following special value can be supplied at startup time: ***EXITPGM** – The actual value will be provided by an exit program later.	CHAR(128)
130	0082	Length of the recipient's name	The length of the data in the following field.	BINARY(2)
132	0084	Recipient's name	The name of the recipient. The following special value can be supplied at startup time: ***EXITPGM** – The actual value will be provided by an exit program later.	CHAR(128)
260	0104	Recipient type	The type of recipient. Possible values are: ***PRI** – Primary ***CC** – CC recipient ***BCC** – BCC recipient	CHAR(10)

			The following special value can be supplied at startup time: ***EXITPGM** – The actual value will be provided by an exit program later.	

Structure CS_EPK01 – Exit program parameters by key (format 1)

Name	Description
CS_EPK01	The CS_EPK01 structure defines options for selecting exit program parameter text from a spooled file by key string.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Page number	The page number in the stream file from which the text should be selected, or the special value: ***-2*** – Select the text from all pages.	BINARY(4)
4	0004	Length of the key string	The length of the following field.	BINARY(2)
6	0006	The key string to find	The key string the occurrence of which is used to select exit program parameter text.	CHAR(512)
518	0206	Occurrence	The occurrence of the key string on the page which is used to select the parameter text.	BINARY(4)
522	020A	Reserved	Reserved	CHAR(4)
526	020E	Horizontal coordinate or column	The horizontal offset from the key string to the start of the data to be selected as a bookmark.	DEC(7,3)
530	0212	Length	The length of the parameter text to select.	DEC(7,3)
534	0216	Unit	The units in which the coordinates and length are defined. ****ROWCOL*** – Rows and columns This is now the only supported option.	CHAR(10)

Structure CS_EPK02 – Exit program parameters by key (format 2)

Name	Description
CS_EPK02	The CS_EPK02 structure defines options for selecting exit program parameter text from a spooled file by key string. Unlike CS_EPK01, all coordinates must be defined in terms of rows and columns, the exact location on the page where the key string should be checked for can be specified, and a vertical offset from the key string to the bookmark string can also be given.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Page number	The page number in the stream file from which the text should be selected, or the special value: ***-2*** – Select the text from all pages.	BINARY(4)
4	0004	Length of the key string	The length of the following field.	BINARY(2)
6	0006	The key string to find	The key string the occurrence of which is used to select exit program parameter text.	CHAR(512)
518	0206	Line number where key should be checked for	The line number on the page where the key string is to be checked for. ***-1*** – Any line. All lines are checked for the key string.	BINARY(4)
522	020A	Column number where key should be checked for	The column number in the line where the key string is to be checked for. ***-1*** – Any column. All columns are checked for the key string.	BINARY(4)
526	020E	Occurrence	The occurrence of the key string on the page which is used to	BINARY(4)

			select the parameter text.	
530	0212	Vertical offset	The vertical offset from the start of the key string to start of the data to be selected as a bookmark, in rows/lines	BINARY(4)
534	0216	Horizontal offset	The horizontal offset from the start of the key string to start of the data to be selected as a bookmark, in columns/characters.	BINARY(4)
538	021A	Length	The length of the parameter text to select, in columns/characters.	BINARY(4)

Structure CS_EPP01 – Exit program parameters by position

Name	Description
CS_EPP01	The CS_EPP01 structure defines options for selecting exit program parameters from a spooled file on a positional basis.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Page number	The page number in the stream file from which the text should be selected, or the special value: ***-2*** – Select the text from all pages.	BINARY(4)
4	0004	Vertical coordinate or line	The vertical coordinate or line number of the text to select as an exit program parameter.	DEC(7,3)
8	0008	Horizontal coordinate or column.	The horizontal coordinate or column of the text to select as an exit program parameter.	DEC(7,3)
12	000C	Length	The length of the parameter text to select.	DEC(7,3)
16	0010	Unit	The units in which the coordinates and length are defined. ****ROWCOL*** – Rows and columns This is now the only supported option.	CHAR(10)

Structure CS_EXT01 – Exit programs

Name	Description
CS_EXT01	The CS_EXT01 structure defines options for calling exit programs.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Exit program name	The name of the exit program to be called.	CHAR(10)
10	000A	Exit program library name	The name of the library in which this exit program is located. You can also use the following special options: ***LIBL** – Locate the exit program through the library list of the current job. ***CURLIB** – The exit program is located in the current library.	CHAR(10)
20	0014	Exit program parameter list type	Defines the parameters which will be passed to the exit program. Options are: ***TYPE1** – Type 1 parameters. ***TYPE2** – Type 2 parameters. ***TYPE3** – Type 3 parameters. ***TYPE4** – Type 4 parameters (to be fully rolled out in the next release) See the section on exit programs for a definition of these parameter lists.	CHAR(10)
30	001E	Exit point	The exit point at which the exit program is to be called,	CHAR(10)

			Options are: ***SPLFSTR** – At the start of processing for the entire spooled file. ***STMFSTR** – At the start of processing for the current stream file. ***PAGESTR** – At the start of processing for the current page. ***SPLFEND** – At the end of processing for the entire spooled file. ***STMFEND** – At the end of processing for the current stream file. ***PAGEEND** – At the end of processing for the current page. ***PAGECTL** – As a page control exit program. This exit point occurs at the start of processing a stream file and prior to processing pages in the file. It provides an opportunity to indicate that certain pages should be excluded from the output (see CS_FBK01).	

Structure CS_FBK01 – Feedback information

Name	Description
CS_FBK01	Feedback information. The CS_FBK01 structure allows an exit program to send feedback information to CoolSpools: specifically, it can send back a flag which determines whether or not CoolSpools includes a specified page or range of pages in the output. This structure can only be generated by an exit program called at the *PAGECTL exit point and is invalid at all other points in the processing of a spooled file.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	No	No	No	No	No	No	Yes

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Page action	A flag determining whether a range of pages is included in the output. The range of pages controlled by this flag is specified in the header structure (scope from-page and scope-to-page fields). To control a single page, specify the from- and to-page values the same. Values are: *1* – Include the page or range of pages in the output. *0* – Exclude the page or range of pages from the output.	CHAR(1)

Structure CS_FNT01 – Font options

Name	Description
CS_FNT01	The CS_FNT01 structure defines various font-related options. It cannot be generated by an exit program.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Font option	The option determining the font used or the way fonts are processed. Select one of the following two font options or one of the standard fonts listed below. ***MAP**– Select an appropriate standard font for fonts used in the spooled file ***EMBED** – Embed fonts (PDF only) ***COURIER** – Courier ***COURIERB** – Courier Bold ***COURIERO** – Courier Oblique ***COURIERBO** – Courier Bold Oblique ***HELVETICA** – Helvetica (in practice the font used may be Arial) ***HELVB** – Helvetica Bold ***HELVO** – Helvetica Oblique ***HELVBO** – Helvetica Bold Oblique ***TIMES** – Times Roman ***TIMESB** – Times Roman Bold ***TIMESI** – Times Roman Italic	CHAR(10)

			***TIMESBI** – Times Roman Bold Italic ***SYMBOL** - Symbol ***DINGBATS** – Zapf Dingbats	
10	000A	Font size	The font size to use. Specify a font size in points or one of the following special values: ***-1*** – Calculate appropriate font sizes based on the font and CPI information held in the spooled ***-2*** – Calculate appropriate font sizes based on the font, CPI and LPI information in the spooled file. This method is the same as the previous except in relation to fonts specified by FGID (font identifier), such as those defined with the DDS FONT keyword In the case of fonts specified by FGID, when the font option is *MAP, the font point size is calculated based on the LPI value and then condensed using a horizontal scaling to the appropriate CPI value. This may give better results in some cases than -1.	BINARY(4)
14	000E	Default font	The font to be used where CoolSpools cannot identify an appropriate font from the information held in the spooled fie. ***COURIER** – Courier ***COURIERB** – Courier Bold ***COURIERO** – Courier Oblique ***COURIERBO** – Courier Bold Oblique ***HELVETICA** – Helvetica (in practice the font used may be Arial) ***HELVB** – Helvetica Bold	CHAR(10)

			***HELVO** – Helvetica Oblique ***HELVBO** – Helvetica Bold Oblique ***TIMES** – Times Roman ***TIMESB** – Times Roman Bold ***TIMESI** – Times Roman Italic ***TIMESBI** – Times Roman Bold Italic ***SYMBOL** - Symbol ***DINGBATS** – Zapf Dingbats	
24	0018	Default font size	The font size to be used where CoolSpools cannot identify the appropriate font size from the information held in the spooled fie.	BINARY(4)
28	001C	Number of font types in the following list	The number of font types in the list that follows.	BINARY(4)
32	0020	List of font types	An array of font types which will be embedded. Fonts not selected by this list will be processed as if the font option had been specified as *MAP. If the font option is not *EMBED, the list must be empty or contain the single value ***NONE**. If the font option is *EMBED, the list can contain the single value: ***ALL** – All types. If the font option is *EMBED, the list can alternatively contain one or more of the following font types: ***CIDKEYED** – CID-keyed DBCS fonts. ***PSTYPE1** – Postscript Type 1 fonts. ***RASTER** – Raster fonts. ***FONTID** – Fonts identified by a	ARRAY 0-10 OF CHAR(10)

			font number (DDS FONT keyword).	

Structure CS_FTP01 – FTP options

Name	Description
CS_FTPR01	The CS_FTP01 structure defines the FTP parameters for use with the special stream file name *FTP.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	Yes	Yes	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of FTP server name	The length of the data in the following field.	BINARY(2)
2	0002	FTP server name	The name or IP address of the FTP server to which the data is to be output by FTP. At startup, the following special value may be specified: ***EXITPGM** – The details will be supplied by an exit program	CHAR(128)
130	0082	Length of FTP path	The length of the data in the following field.	BINARY(2)
132	0084	Server path	The server path where the output is to be saved. This is the full path including the file name. ***EXITPGM** – The details will be supplied by an exit program	CHAR(256)
388	0184	Port number	The port number to use when connecting to the server. ***-1*** - The details will be supplied by an exit program	BINARY(4)
392	0188	Length of the user id	The length of the data in the following field.	BINARY(2)
394	018AB	User id	The logon user id to be used to	CHAR(128)

			connect to the server ***EXITPGM** – The details will be supplied by an exit program	
522	020A	Length of the password	The length of the data in the following field.	BINARY(2)
524	020C	Password	The logon password to be used to connect to the server ***EXITPGM** – The details will be supplied by an exit program	CHAR(128)

Structure CS_HTM01 – HTML options

Name	Description
CS_HTM01	HTML-related options

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of script path	The length of the script path defined in the following field	BINARY(2)
2	0002	Script name	A path defining the name of a stream file. CoolSpools will retrieve the contents of this file and embed them in the HTML file header. This is intended to allow you to include items such as JavaScript code in your generated HTML file.	DEC(7,3)

Structure CS_INC01 – Included images

Name	Description
CS_INC01	The CS_INC01 structure allows you to define a JPEG image to be included in a PDF.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of path	The length of the value defined in the following field	BINARY(2)
2	0002	Path	The IFS path where the file to be included can be located.	CHAR(256)
258	0102	Format	The format of file. Possible values are: ***JPG** – JPEG ***GIF** – GIF	CHAR(10)
268	010C	Method	Possible values are: ***EMBEDDED** – The JPEG file is embedded in the PDF file. This will increase the size of the PDF but guarantees that the JPEG will be available when the PDF file is opened. ***EXTERNAL** – The JPEG file is included by means of a reference to an external file. This will minimize the size of the PDF file but you must ensure that the JPEG can be retrieved from the specified location when the PDF file is opened.	CHAR(10)
278	0116	Where included	The pages on which the file is to be included.	CHAR(10).

			Possible options are: ***ALL*** – All pages ***ODD*** – Odd-numbered pages only ***EVEN*** – Even-numbered pages only ***FIRST*** – The first page only ***LAST*** – The last page only ***AFTFIRST*** – All pages after the first page ***BFRLAST*** – All pages before the last page ***BACK*** –An extra page is inserted after each page in the spooled file and the image is included on this extra page. This option is useful where you have a pre-printed form with information printed on the reverse. You can include this information in the PDF file on an additional page by using this option. ***FRONT*** – As with *BACK, an extra page is inserted after each page in the spooled file. Unlike *BACK, the image is included on the original page, not the inserted page. ***HEADER*** - An extra page is inserted at the start of each group of pages which forms a single PDF file and the image is included on this additional page. This option can be useful if you wish to have a header sheet at the start of the PDF file. ***TRAILER*** - An extra page is inserted at the end of each group of pages which forms a single PDF file and the image is included on this additional page. This option can be useful if you wish to have a trailer sheet at the end of the PDF file	

			***KEYABS** - The image is included if the key string ("Key string" parameter below) occurs on the page. The coordinates are interpreted as absolute coordinates. ***KEYREL** - The image is included if the key string ("Key string" parameter below) occurs on the page. The coordinates are interpreted as relative coordinates, relative to the key string.	
288	0120	X coordinate	The horizontal position on the page where the top-left corner of the image should be placed This is interpreted as an absolute position on the page unless a key string is specified, in which case (unless “Included on pages” is *KEYABS), this is interpreted as relative to the start of the key string.	DEC(7,3)
292	0124	Y coordinate	The vertical position on the page where the top-left corner of the image should be placed This is interpreted as an absolute position on the page unless a key string is specified, in which case (unless “Included on pages” is *KEYABS), this is interpreted as relative to the start of the key string.	DEC(7,3)
296	0128	Unit	The unit in which these coordinates are measured: ***INCH** – inches ***CM** – centimeters ***MM** – millimeters	CHAR(10)
306	0132	Length of external reference	The length of the value in the following field	BINARY(2)
308	0134	External	The path to be included in the	CHAR(256)

		reference	PDF file and which Acrobat will use to locate the JPEG file when the PDF is opened	
564	0234	External reference type	The type of external reference Possible values are: ***DOS** – A DOS/Windows-style path name ***MAC** – A Mac -style path name ***UNIX** – A UNIX -style path name ***URL** – A URL	CHAR(10)
574	023E	Scaling factor	The scaling factor to apply to increase or reduce the apparent size of the image. A value of 1 will display the image actual size.	DEC(7,3)
578	0242	Rotation	The angle by which to rotate the image	DEC(7,3)
582	0246	Length of key string	The length of the data in the following field.	BINARY(2)
584	0248	Key string	If "Include on pages" is not *KEYABS or *KEYREL, the X and Y coordinates defined above are interpreted as offsets relative to the start of this key string. If "Include on pages" is *KEYREL, the image only appears if the key string occurs on the page and the X and Y coordinates defined above are interpreted as offsets relative to the start of this key string. If "Include on pages" is *KEYABS, the image only appears if the key string occurs on the page and the X and Y coordinates defined above are interpreted as absolute coordinates on the page.	CHAR(512)
1096	0448	Occurrence	Which occurrence of the key string on the page determines the positioning of the image. If the key string occurs more than	BINARY(4)

			once on the page, you can specify which occurrence to use on this parameter.	
1100	044C	Key action	Whether the key string is included in the output or deleted. If you have included the key string in the spooled file simply to indicate the location where an image should be positioned, you can ensure that it is not visible in the final PDF file by telling CoolSpools to remove it. Options are: ***KEEP** - Keep the key string in the output. ***REMOVE** - Remove the key string from the output.	CHAR(10)

Structure CS_LIC01 – License information

Name	Description
CS_LIC01	License information. This structure is not intended for use by customers who have purchased a standard CoolSpools license. It is reserved for use by companies who have licensed CoolSpools functionality for inclusion in their own licensed programs and who wish to perform their own license key checking. If a valid product code, password and caller id are specified here, CoolSpools will not carry out its own license checks and will leave the responsibility for checking the customer license to the calling application.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Product code	The product id of the calling licensed program.	CHAR(7)
7	0007	Product password	The password you were given by ariadne when you licensed the code for inclusion in your product.	CHAR(32)
39	0027	Caller id	A code which will be supplied to you by ariadne when you license the code. This indicates the functions from which CoolSpools is being called and determines which CoolSpools functions are available.	CHAR(20)

Structure CS_MGN01 – Margins

Name	Description
CS_MGN01	The CS_MGN01 structure defines PDF margin and positioning options.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Unit	The unit in which the following options are defined: ***INCH*** – inches ***CM*** – centimeters ***MM*** - millimeters	CHAR(10)
10	000A	Additional left margin	The margin to be added to the left of the page	DEC(7,3)
14	000E	Additional top margin	The margin to be added at the top of the page	DEC(7,3)
18	0012	Overlays left margin	The margin to be added to the left of overlay contents ***-999*** – Calculate the appropriate value.	DEC(7,3)
22	0016	Overlays top margin	The margin to be added at the top of overlay contents. ***-999*** – Calculate the appropriate value.	DEC(7,3)
26	001A	Text left shift	The amount by which text should be shifted from the left edge of the page ***-999*** – Calculate the appropriate value.	DEC(7,3)

30	001E	Text top shift	The amount by which text should be shifted from the top edge of the page ***-999*** – Calculate the appropriate value.	DEC(7,3)
34	0022	Increase page size	Whether margins should be created by increasing the page size. ****YES*** – Increase the page size ****NO*** – Do not increase the page size	CHAR(10)

Structure CS_OPT01 – Miscellaneous options

Name	Description
CS_OPT01	Miscellaneous options. The CS_OPT01 controls various options which do not belong naturally to any other option structure.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	Yes	Yes	Yes	Yes	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Title	The file title. For PDF, this appears in the Document Information. For HTML, this appears in the browser title. Specify the title text or one of the following special values: **NONE* – The file will have no title. **STMFILE* – The file will have its title derived from the stream file name.	CHAR(50)
50	0032	Include overlay data	A code controlling whether text derived from an overlay or page segment object is included in the output. Overlays and page segments often contain constants, labels and heading text while the spooled file itself contains the variable data associated with those constants, labels and headings. For example, you might have an invoicing application where your overlay contains text such as "Customer name", "Invoice number" and "Invoice date" and your spooled file supplies the actual customer name, invoice	CHAR(10)

			number and invoice date information to be printed alongside those labels. In some circumstances, for example when creating PDF versions of your spooled file, it may be appropriate to include the overlay text for the sake of clarity. However, in other circumstances, for example when converting the data to ASCII text format for indexing purposes, it might be more appropriate to exclude the overlay text and just process the variable data from the spooled file itself. This option allows you to indicate which option you wish to choose. Values are: **TOFMT* - CoolSpools determines whether to include overlay text based on the format of the stream file being created. Overlay text is included if the spooled file is being converted to *PDF or *HTMLCSS, otherwise it is excluded. **SPLF* - CoolSpools determines whether to include overlay text based on the format of the spooled file being converted. Overlay text is excluded if the spooled file is being converted is *SCS, otherwise it is included. **YES* - Overlay text is included in the stream file. **NO* - Overlay text is excluded from the stream file. **OUTPUT* - Overlay text is included in the stream file. However, text from overlays and page segments is not included in processing when text functions such as bookmarks, split triggers and exit program parameters are	

			being generated. **IGNOVLDTA* - All content from overlay and page segment objects (both text and non-text) is ignored and dropped from the processing.	
60	3C	Column calculation method	This parameter controls the way in which CoolSpools calculates columns in the report, for the purposes of text selection (e.g. bookmarks, split keys, exit program parameters). Where the spooled file contains text with different font sizes, especially proportional fonts, it is not obvious how to calculate the text "column" for a piece of text when text is being selected using the *ROWCOL method. Values are: **CPI* - CoolSpools calculates text columns using the CPI attribute of the spooled file. This is the method used by DSPSPLF. This is now the recommended value but the following value can be used if you need to ensure that applications continue to behave as under earlier versions of CoolSpools. **FONT* - CoolSpools calculates text columns using the different font sizes in the spooled file.	CHAR(10)
70	0046	Codepage	The encoding to use for data in the stream file. The rules used for deciding how system i data should be converted are complex, and depend on numerous factors, including the stream file format being created (as specified on the TOFMT parameter). In particular, PDF implements its own rules with regard to data encoding and these are followed in all	BINARY(4)

			instances. The value specified here is therefore ignored in relation to PDF creation and CoolSpools will convert the spooled file contents and all other data (such as the values specified on the TITLE or KEYWORDS parameters) to a format compatible with PDF. Specify a codepage number or use one of the special values: *0* – CoolSpools calculates the most appropriate stream file codepage based on the spooled file CCSID and to-format. *-1* -.Where appropriate, a suitable Windows ASCII encoding scheme is selected. For example, when converting a spooled file to *TEXT on a US system i (CCSID 37), code page 1252 (Windows Western ASCII encoding) is selected for the ASCII text, whereas on a Greek system i (CCSID 875), code page 1253 (Windows Greek) is selected. *-2* - Where appropriate, a suitable IBM PC encoding scheme is selected. For example, when converting a spooled file to *TEXT on a US system i (CCSID 37), code page 437 (IBM PC ASCII) is selected for the ASCII text, whereas on a Greek system i (CCSID 875), code page 869 (IBM PC Data Greek) is selected. *-3* - Where appropriate, a suitable ISO ASCII encoding scheme is selected. For example, when converting a spooled file to *TEXT on a US system i (CCSID 37), code page 819 (ISO 8859-1) is selected for the ASCII text, whereas on a Greek system i (CCSID 875), code page 813 (ISO 8859-7) is selected. *-4* - If the stream file exists, and	

			where appropriate, the code page of the existing stream file is used for data conversion purposes. *-5* - Where appropriate, the data is converted to Unicode format. Note that Unicode data produced by CoolSpools is in big-endian format (i.e. the most significant byte is stored first, according to the system i convention, as opposed to the PC little-endian convention, which stores the least significant byte first). When opening a text file created by CoolSpools with STMFCODPAG(*UNICODE) specified in an application such as Windows Notepad, select the Unicode big-endian format option. -6 Reserved for future use. -7 No conversion occurs. The data is output in the same format in which it is held in the original spooled file. This may give the best results with certain languages such as Arabic.	
74	4A	Spooled file CCSID	The CCSID (Coded Character Set Identifier) which CoolSpools should assume when converting the data content of the spooled file, in the absence of any other indication of the appropriate CCSID to use. The CCSID specifies the encoding scheme used to represent the data and determines how particular code point values will be interpreted and converted in the stream file that is to be created. Although more sophisticated printer data streams such as AFP and IPDS will include information which indicate the encoding scheme used to represent data in the spooled file, SCS spooled files	BINARY(4)

			often contain no explicit information to allow CoolSpools to determine the CCSID of the data. Although the CHRID value of the printer file may, in some cases, make it possible to interpret the CCSID to be used, in many instances the CHRID value will be *DEVD, meaning that the character identifier attribute of the printer device used to print the spooled file will determine the character set. CoolSpools cannot know which printer device was intended to print the spooled file and will assume that the CCSID to use is the CCSID indicated by the QCCSID or QCHRID system values. If the spooled file has CHRID(*DEVD) specified and was intended to be printed on a printer which uses a character set which is different from that implied by the QCCSID or QCHRID system values, it will be necessary to specify the appropriate CCSID on this parameter in order for CoolSpools to interpret the spooled file contents appropriately. You may specify a CCSID or use one of the special values: *-1* - The value of the QCCSID system value is used, or a CCSID derived from the QCHRID system value is used if QCCSID is 65535. -2 - The CCSID of the current job is used. If the CCSID of the job is 65535, the default CCSID attribute of the job is used. *-3* - CoolSpools will use the information available from the spooled file to determine the correct CCSID to use. As explained above, if the spooled file does not contain adequate	

			information relating to the encoding of the data it contains, CoolSpools will assume the CCSID implied by the QCCSID or QCHRID system values. *-4* - The value of the CCSID attribute of the user profile of the user running the command is used.	
78	4E	Exit program parameter CCSID	The CCSID in which the parameters will be passed to the exit program. If your spooled file contains ASCII data (for example, if it is a PCL *USERASCII spooled file), you may wish to have CoolSpools convert the data to a more convenient CCSID before passing it to your program. You may specify a CCSID or use one of the special values: *-1* - The value of the QCCSID system value is used, or a CCSID derived from the QCHRID system value is used if QCCSID is 65535. -2 - The CCSID of the current job is used. If the CCSID of the job is 65535, the default CCSID attribute of the job is used. *-3* - CoolSpools will use the information available from the spooled file to determine the correct CCSID to use. As explained above, if the spooled file does not contain adequate information relating to the encoding of the data it contains, CoolSpools will assume the CCSID implied by the QCCSID or QCHRID system values. *-4* - The value of the CCSID attribute of the user profile of the user running the command is used. *-5* – The CCSID attribute of the	BINARY(4)

			report definition is used.	
82	52	Authority	The public authority given to the stream file when it is created. Specify one of the following values: **R* - Read only **W* - Write only **X* - Execute only **RW* - Read and write **RX* - Read and execute **WX* - Write and execute **RWX* - Read, write and execute (all) **NONE* - No authority Alternatively, specify the name of an authorization list. This authorization list will be associated with the stream file and authorities for *PUBLIC assigned from the authorization list.	CHAR(10)
92	5C	Inherit authority.	Allows you to control whether object authorities are inherited from the parent directory in which the stream file is created. **YES* - Authorities are inherited from the directory. **NO* - Authorities are not inherited from the directory When you specify *NO, the object authorities (*OBJEXIST, *OBJMGT, *OBJALTER, and *OBJREF) assigned to the owner, primary group, and *PUBLIC in respect of the stream file being created are copied from the owner, primary group, and public object authorities of the parent directory in which the stream file is created. This occurs even when the new file has a different owner	CHAR(10)

			from the parent directory. The new file does not have any private authorities or authorization list. It only has authorities for the owner, primary group, and public. The owner is assigned full data authorities and *PUBLIC is assigned the data authorities specified in the previous field. When you specify *YES, the object authorities (*OBJEXIST, *OBJMGT, *OBJALTER, and *OBJREF) assigned to the owner, primary group, and *PUBLIC in respect of the stream file being created are copied from the owner, primary group, and public object authorities of the parent directory in which the stream file is created. However, the private authorities (if any) and authorization list (if any) are also copied from the parent directory. If the new file has a different owner from the parent directory and the new file's owner has a private authority in the parent directory, that private authority is not copied from the parent directory. The authority for the owner of the new file is copied from the owner of the parent directory. The owner is assigned full data authorities and *PUBLIC is assigned the data authorities specified on the AUT parameter. *YES is now the recommended value unless you need CoolSpools to operate as it did by default in previous versions.	
102	0066	Include blank lines	Whether blank lines should be included in the output for formats *TEXT, *HTXT and *HTML. ***YES** = Include blank lines ***NO** = Exclude blank lines ***FF** = Include blank lines and	CHAR(10).

			insert a formfeed character to indicate the end of a page.	
112	0070	Line calculation method	This parameter controls the way in which CoolSpools calculates line numbers in the report, for the purposes of creating text files and for text selection (e.g. bookmarks, split keys, exit program parameters). Where the spooled file contains text with different font sizes, especially proportional fonts, it is not obvious how to calculate the text "line" for a piece of text when text is being selected using the *ROWCOL method. This field is optional. If the structure is supplied without it (i.e. its length is less than 122 bytes), it defaults to *OLD. Values are: ***NEW** - CoolSpools calculates text columns using the LPI attribute of the spooled file. This is the method used by DSPSPLF. This is now the recommended value but the following value can be used if you need to ensure that applications continue to behave as under earlier versions of CoolSpools. This may be necessary where you have applications which have line numbers coded and you need these to continue to select the same data in your reports. ***OLD** - CoolSpools calculates text columns using a slightly different method. This is the method used by versions prior to Version 6 and prior to the application of Version 6 PTF 5CV0028. ***ENVVAR** - CoolSpools uses the method indicated by environment variable CS_TXT_LINE_METHOD. If this exists, and is set to *NEW, the	CHAR(10)

			new method is used, otherwise the old method is used. A job-level environment variable overrides a system-level environment variable.	
122	007A	CPI value	Characters Per Inch setting to use or assume when performing text conversions. This includes outputting to text format (e.g. CVTSPLTXT or CVTSPLCSV) but also calculation of bookmarks, exit program parameters, split strings etc. You may specify a characters-per-inch setting or use one of the following special values: ***-1*** = (*SPLF) Use spooled file attribute ***-2*** = (*BESTFIT) CoolSpools calculates a value based on the smallest font size used in the file, intended to avoid text overlaying other text in the output. ***-3*** = (*RPTDFN) The CPI setting associated with the report definition is used.	DEC(3,1)
124	007C	LPI value	Lines Per Inch setting to use or assume when performing text conversions. This includes outputting to text format (e.g. CVTSPLTXT or CVTSPLCSV) but also calculation of bookmarks, exit program parameters, split strings etc. You may specify a characters-per-inch setting or use one of the following special values: ***-1*** = (*SPLF) Use spooled file attribute ***-2*** = (*BESTFIT) CoolSpools calculates a value based on the smallest font size used in the file, intended to avoid text overlaying	

			other text in the output. ***-3*** = (*RPTDFN) The LPI setting associated with the report definition is used.	
126	007E	Report definition name	Report definition name. This must be the name of the report definition associated with the report map specified below.	CHAR(20)
146	0092	Report map type	The type of report map specified below. Must be one of: ***XLS*** = Report-to-Excel map ***XML*** -= Report-to-XML map ***DBF*** = Report-to-database map	CHAR(3)
149	0095	Report map name	The report map to use. The report definition associated with this map must be the one specified above.	CHAR(20)
169	00A9	Reserved	Reserved	BINARY(4)
173	00AD	Reserved	Reserved	BINARY(4)
177	00B1	Unicode endianness	Whether multi-byte Unicode values are output in bigendian or littleendian format: Options are: ***B*** – Bigendian ***L*** - Littleendian	CHAR(1)
178	00B2	Unicode marker	Whether or not to output a Unicode marker at the beginning of a Unicode text file indicating the type of Unicode encoding used. Options are: ***N*** – No, do not output a marker ***Y*** – Yes, output a marker	CHAR(1)
179	00B3	Text line numbers	Whether or not to output a line numbers at the beginning of each line of a text file. Options are:	CHAR(1)

			N – No, do not output line numbers ***Y*** – Yes, output line numbers	
180	00B4	Text column numbers	Whether or not to output column numbers at the beginning of a text file. Options are: ***N*** – No, do not output column numbers ***Y*** – Yes, output column numbers	CHAR(1)

Structure CS_PDF01 – PDF options

Name	Description
CS_PDF01	PDF options.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Bookmark option	The type of bookmarks required. Any attempt to override this value in an exit program will be ignored. ***NONE*** – No bookmarks. ***PAGNBR*** – Page number bookmarks ***POS*** – Positional bookmarks defined by CS_BMP01 structures. ***KEY*** – Key bookmarks defined by CS_BMK01 structures. ***POSKEY*** – Positional and key bookmarks defined by CS_BMP01 and CS_BMK01 structures. ***PAGIDXTAG*** – Bookmarks generated from page-level index tags in the spooled file ***GRPIDXTAG*** – Bookmarks generated from group-level tags in the spooled file ***IDXTAG*** – Bookmarks generated from page- and group-level tags in the spooled file.	CHAR(10)
10	000A	Bookmark action	The initial bookmark action: ***SHOW*** – Show bookmarks ***HIDE*** – Hide bookmarks	CHAR(10)
20	0014	Viewer type	The type of viewer for which the	CHAR(10)

			PDF file is intended: ***WINDOWS** – A Windows-based viewer such as Acrobat Reader for Windows. ***OTHER** – A viewer intended for a different platform (e.g. Mac or LINUX)	
30	001E	Zoom factor	The initial zoom factor. This can be a percentage between 8.33 and 1600, where 100 = actual size, or one of the following special values: ***-1*** - PDF default (the default zoom factor for your browser) ***-2*** – Fit window ***-3*** – Actual size ***-4*** – Fit width ***-5*** – fit visible	DEC(7,2)
34	0022	Data compression	***OPT** - Stream data in PDF files is compressed. The level of compression that is applied provides a good degree of data compression while not taking unduly long to compress. ***NONE** - Stream data in PDF files is not compressed. The resultant PDF files will be significantly larger than if data compression was applied, but will take less time to create. The following additional options are available, in decreasing order of data compression ratio and increasing order of speed to process. ***MAXIMUM*** ***HIGHER*** ***.*HIGH*** ***FAST*** ***FASTER**. ***FASTEST***	CHAR(10)

44	002C	Fast Web View	Determines whether CoolSpools generates a PDF file with the Fast Web View option enabled to speed up opening of the file across a network. ***YES** – Fast Web View enabled ***NO** – Fast Web View disabled	CHAR(10)
54	0036	Show rotated pages un-rotated	Determines whether rotated pages are displayed in their rotated form or in the natural orientation of reading. ***YES** - Where CoolSpools has applied a page rotation, the page will be rotated back into the natural orientation for reading. ***NO** – Where CoolSpools has applied a page rotation, the page will be displayed in its rotated form.	CHAR(10)
64	0040	Keywords for indexing	This element allows you to define a set of keywords to be included in the Document Info section of the PDF file. These can be used by indexing and document management applications. Specify the keywords as a single character string with individual keywords separated by a comma or semicolon.	CHAR(256)

Structure CS_PGO01 – Page Options

Name	Description
CS_PGO001	Page options. Defines various options related to the presentation of data on the page.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Auto-rotation	Whether an automatic rotation, and/or Computer Output Reduction (COR) are applied, simulating the effects of the PAGRTT(*AUTO), PAGRTT(*COR) or PAGRTT(*DEVD) attribute on certain printers. If your spooled file has the attribute PAGRTT(*AUTO), PAGRTT(*COR) or PAGRTT(*DEVD), automatic page rotation will occur when the spooled file is printed on a printer and the spooled file does not fit on the page in its standard orientation. For example, if you have a spooled file which is in landscape mode, and you print it on a printer which prints in A4 portrait mode normally, the spooled file will be rotated automatically through 90 degrees and printed in landscape mode on the A4 portrait printer. In addition, if PAGRTT(*COR) is specified, or if the spooled file still does not fit on the page even after rotation, Computer Output Reduction (COR) may be applied, subject to the type and attributes	CHAR(10)

			of the spooled file fulfilling certain conditions. COR involves scaling the contents of the spooled file horizontally and vertically to fit on an 8.5 x 11 inch sheet of paper. If you are relying on a page rotation, it is possible that page segments and images contained in the spooled file will appear misplaced and rotated in the PDF files you create using the default CoolSpools parameters. In order to replicate the effects of an automatic page rotation in your spooled file, you need to specify *YES here. This will ensure that images and page segments are handled as if the page had rotated. If you are not sure if rotation and COR are appropriate, you can specify *SPLF, in which case CoolSpools will attempt to predict the behaviour of the most modern printers, and will itself decide whether to rotate the page or apply COR. CoolSpools will assume that the paper size specified on the first two elements of this parameter indicate the paper size on which the document is normally printed, and will decide whether rotation and/or COR are required based on this paper size. Please note that the paper size specified in the CS_PGS01 structure is interpreted as the shape and format of the paper prior to rotation. For example, if your document prints on cut sheet letter paper (11 x 8.5 inches), you should specify *LETTER for the paper size and *PORTRAIT for the orientation even if the document prints in landscape mode, since the paper is physically printed in portrait mode	

			and the document contents rotated to fit on it. **CALC* – Calculate whether to apply an auto-rotation based on the spooled file attributes and other options. **SPLF* – Same as *CALC. Provided for consistency with previous versions. **YES* – Assume an auto-rotation **NO* – Do not assume an auto-rotation. **COR* - Auto-rotation and COR will both be applied. **PAGESIZE* - Auto-rotation is applied if the spooled file attributes suggest a landscape orientation (i.e. if the calculated page width is larger than the calculated page length), but not if the attributes suggest a portrait orientation (width less than height).	
10	000A	Horizontal scaling	Specify the horizontal scaling as a number between 0.01 and 99.99, or specify one of the following special values: 1 – No scaling -1 – Calculate the appropriate scaling based on the spooled file attributes and other parameters. -2 – Calculate a scaling factor to fit the spooled file contents to the page size selected.	DEC(4,2)
13	000D	Vertical scaling	Specify the vertical scaling as a number between 0.01 and 99.99, or specify one of the following special values: *1* – No scaling *-1* – Calculate the appropriate scaling based on the spooled file attributes and other parameters. *-2* – Calculate a scaling factor to	DEC(4,2)

			fit the spooled file contents to the page size selected.	

Structure CS_PGS01 – Page Size information

Name	Description
CS_PGS001	Page size information. Defines the dimensions and orientation of the paper to be targeted.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Paper size	The page size to be targeted. **CALC* – CoolSpools will select a page size based on the system settings. This is normally equivalent to *LETTER in the US and *A4 elsewhere. **SPLF* - The dimensions are derived from the spooled file attributes. **DEVD* – The page size is determined from the attributes of the device specified in the CS_DEV01 structure. **CUSTOM* - The page size is a non-standard size and is defined in the CS_CST01 structure. **A3* **A4* **A5* **B4* **B5* **LETTER* **LEGAL* **EXEC* **LEDGER*	CHAR(10)

10	000A	Page orientation	**PORTRAIT* – Portrait orientation **LANDSCAPE* – Landscape orientation	CHAR(10)

Structure CS_PWD01 – Password options

Name	Description
CS_PWD01	The CS_PWD01 structure defines password and security options for PDF files.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	Yes	Yes	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Password option	Whether security is applied to PDF files and, if so, how. Values are: ***YES** – At least one password will be defined for the PDF file. ***NO** – No passwords or security will be applied to the PDF file. ***RESTRICT** – No passwords will be defined for the PDF file but the operations which can be applied to it will be restricted. ***EXITPGM** – Passwords will be supplied by an exit program.	CHAR(10)
10	000A	Length of the user password	The length of the data in the following field. Must be zero if the password option is not *YES. Must not be zero if the password option is *YES and the owner password length is zero.	BINARY(2)
12	000C	User password	The user password to be applied to the file. When the file is opened, the user will be prompted to enter a password. Entering this password will allow the file to be opened, but any restrictions specified will be in effect.	CHAR(32)

			If no user password is provided, and the password option is *YES, an owner password must be provided. Providing an owner password but no user password means that no password will be prompted for when the file is opened in Acrobat Reader. However, opening the file in Acrobat and supplying the owner password will give unrestricted access to the file.	
44	002C	Length of the owner password	The length of the data in the following field. Must be zero if the password option is not *YES. Must not be zero if the password option is *YES and the user password length is zero.	BINARY(2)
46	002E	Owner Password	The owner password to be applied to the file. If the file has a user password, when it is opened in Acrobat Reader the user will be prompted to enter a password. Entering the owner password will open the file and give unrestricted access to it. If no owner password is provided, and the password option is *YES, a user password must be provided. Providing a user password but no owner password means that there will be no means of gaining unrestricted access to the file.	CHAR(32)
78	004E	Allow printing	Whether the file may be printed. This field must be blank if the password option is *NO. If the password option is *YES, possible values are: ***YES*** – Printing is allowed ***NO*** – Printing is not allowed	CHAR(10)

88	0058	Allow modification	Whether the file may be modified. This field must be blank if the password option is *NO. If the password option is *YES, possible values are: ***YES** – Modifications are allowed ***NO** – Modifications are not allowed	CHAR(10)
98	0062	Allow copying of text	Whether text in the file may be copied. This field must be blank if the password option is *NO. If the password option is *YES, possible values are: ***YES** – Copying of text is allowed ***NO** – Copying of text is not allowed	CHAR(10)
108	006C	Allow annotation	Whether the file may be annotated. This field must be blank if the password option is *NO. If the password option is *YES, possible values are: ***YES** – Annotation is allowed ***NO** – Annotation is not allowed	CHAR(10)

Structure CS_RSC01 – Resource directory

Name	Description
CS_RSC01	Resource directory options

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of resource directory name	The length of the name specified in the following field	BINARY(2)
2	0002	Resource directory name	The name of the directory where CoolSpools will look for saved resources such as PCL macros and soft fonts.	CHAR(*)

Structure CS_RTF01 – RTF options

Name	Description
CS_RTF01	RTF-related options

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Left margin	The left margin to apply to the page.	DEC(7,3)
4	0004	Top margin	The top margin to apply to the page.	DEC(7,3)
8	0008	Right margin	The right margin to apply to the page.	DEC(7,3)
12	000C	Bottom margin	The bottom margin to apply to the page.	DEC(7,3)
16	0010	Unit	The unit in which the margins are defined. ****INCH*** – inches ****CM*** – centimeters ****MM*** - millimeters	CHAR(10)
26	001A	Spacing	The spacing to be used between paragraphs in the RTF document, measured in points. A point is (approximately) 1/72 of an inch. Specify a spacing value in points or the following special value: ***-998 -*** Spacing between paragraphs in the RTF document is calculated based on the page size being targeted to ensure that the text appears on the page in the same position it did in the original spooled file, irrespective	BINARY(4)

			of any changes of font size applied. ***-999*** **-** Spacing between paragraphs in the RTF document is calculated based on the spooled file attributes to ensure that the text appears on the page in the same position it did in the original spooled file, irrespective of any changes of font size applied.	

Structure CS_SAV01 – *SAV options

Name	Description
CS_SAV01	Options for use with TOFMT(*SAV), which creates a spooled file archive.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Data compression	***OPT** - The level of compression that is applied provides a good degree of data compression while not taking unduly long to compress. ***NONE** – No compression is applied. The following additional options are available, in decreasing order of data compression ratio and increasing order of speed to process. ***MAXIMUM*** ***HIGHER*** ***.*HIGH*** ***FAST*** ***FASTER**. ***FASTEST***	CHAR(10)

Structure CS_SPK01 – Split by key options (format 1)

Name	Description
CS_SPK01	The CS_SPK01 structure defines options for splitting a spooled file by key string.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of the key string	The length of the data in the following field.	BINARY(2)
2	0002	Key string	The key string which triggers a split when it fulfills the criteria.	CHAR(512)
514	0202	Method	The split by key method. Possible values are: ***ALWAYS** - Splitting occurs every time the key string occurs in the file. ***IF** - When the key string appears in the file, the following condition is tested and splitting occurs if the condition if true.	CHAR(10)
524	020C	Reserved	Reserved.	CHAR(4)
528	0210	Horizontal offset	The offset from the start of the key string to start of the data to be tested.	DEC(7,3)
532	0214	Length	The length of the data to be tested.	DEC(7,3)
536	0218	Unit	The units in which the coordinates and length are defined. ***ROWCOL** – Rows and columns This is now the only supported option.	CHAR(10)

546	0222	Operator	The operator to use when performing the comparison. Possible values are: ***EQ** – Equal to ***NE** - Not equal to ***GT** – *Greater than* ***LT** – Less than ***GE** - Greater than ***LE** - Less than or equal to ***CT** - Contains ***NC** – Does not contain	CHAR(10)
556	022C	Length of the comparison string	The length of the data in the following field.	BINARY(2)
558	022E	Comparison string	The string with which the selected data is to be compared, or the special value: ***PRV** – The data at this position on the previous page.	CHAR(50)

Structure CS_SPK02 – Split by key options (format 2)

Name	Description
CS_SPK02	The CS_SPK02 structure defines options for splitting a spooled file by key string. Unlike CS_SPK01, all coordinates must be defined in terms of rows and columns, the exact location on the page where the key string should be checked for can be specified, and a vertical offset from the key string to the bookmark string can also be given.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of the key string	The length of the data in the following field.	BINARY(2)
2	0002	Key string	The key string which triggers a split when it fulfils the criteria.	CHAR(512)
514	0202	Line number where key should be checked for	The line number on the page where the key string is to be checked for. ***-1*** – Any line. All lines are checked for the key string.	BINARY(4)
518	0206	Column number where key should be checked for	The column number in the line where the key string is to be checked for. ***-1*** – Any column. All columns are checked for the key string.	BINARY(4)
522	020A	Method	The split by key method. Possible values are: ****ALWAYS*** - Splitting occurs every time the key string occurs in the file. ****IF*** - When the key string appears in the file, the following condition is tested and splitting occurs if the	CHAR(10)

			condition if true.	
526	020E	Vertical offset	The vertical offset from the start of the key string to start of the data to be selected as a bookmark, in rows/lines	BINARY(4)
530	0212	Horizontal offset	The horizontal offset from the start of the key string to start of the data to be selected as a bookmark, in columns/characters.	BINARY(4)
534	0216	Length	The length of the data to be tested, in columns/characters.	BINARY(4)
538	022A	Operator	The operator to use when performing the comparison. Possible values are: ***EQ** – Equal to ***NE** - Not equal to ***GT** – *Greater than* ***LT** – Less than ***GE** - Greater than ***LE** - Less than or equal to ***CT** - Contains ***NC** – Does not contain	CHAR(10)
548	0224	Length of the comparison string	The length of the data in the following field.	BINARY(2)
550	0226	Comparison string	The string with which the selected data is to be compared, or the special value: ***PRV** – The data at this position on the previous page.	CHAR(50)

Structure CS_SPL01 – Spooled File options

Name	Description
CS_SPL01	Spooled file options. Allows an exit program to specify various options relating to the creation of spooled files from an original spooled file (as with the CVTSPLSPLF command).

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	Yes	No	No

.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	New owner	The owner of the new spooled file(s). Specify the user profile that should own the new spooled files or one of the following special values: ***CURRENT** The spooled files are owned by the user running the command. ***SPLF** The owner should be the same as the owner of the original spooled file. If the owner's user profile does not exist on the system, an error will occur.	CHAR(10)
10	000A	Output priority	The output priority to be assigned to the spooled file when it is created. Specify the priority number (1-9) or one of the following special values: **J** - The output queue priority defined by the OUTPTY attribute of the job running the command is used. **S** - The output priority of the original spooled file is used.	CHAR(1)

			However, if this output priority exceeds the maximum output priority allowed for the user who is restoring the spooled file, the restore operation will fail. This error can be avoided by specifying a different (lower) output priority on this parameter.	
11	000B	Output queue	When creating spooled files from an original spooled file, this option defines the output queue on which the new spooled files should be created. Specify the name of the output queue or one of the following special values: ***JOB** The output queue defined by the OUTQ attribute of the job running the command is used. ***SPLF** The output queue on which the original spooled file is located is used.	CHAR(10)
21	0015	Outq library	Specify the name of the library in which the output is located or one the following special values: ***LIBL** - The output queue should be located through the library list of the current job. ***CURLIB** - The output queue is located in the current library of the job that is calling the API.	CHAR(10)

Structure CS_SPP01 – Split by position options

Name	Description
CS_SPP01	The CS_SPP01 structure defines options for splitting a spooled file on a positional basis.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Vertical coordinate or line number	The vertical coordinate or line number of the data to be checked.	DEC(7,3)
4	0004	Horizontal coordinate or column	The horizontal coordinate or column of the start of the data to be checked.	DEC(7,3)
8	0008	Length	The length of the data to be checked	DEC(7,3)
12	000C	Unit	The units in which the coordinates and length are defined. ***ROWCOL** – Rows and columns This is now the only supported option.	CHAR(10)
22	0016	Operator	The operator to use when performing the comparison. Possible values are: ***EQ** – Equal to ***NE** - Not equal to ***GT** – *Greater than* ***LT** – Less than ***GE** - Greater than ***LE** - Less than or equal to ***CT** - Contains	CHAR(10)

			***NC** – Does not contain	
32	0020	Length of the comparison string	The length of the data in the following field.	BINARY(2)
34	0022	Comparison string	The string with which the selected data is to be compared, or the special value: ***PRV** – The data at this position on the previous page.	CHAR(50)

Structure CS_SPT01 – Splitting options

Name	Description
CS_SPT01	The CS_SPT01 structure defines options for splitting a spooled file into multiple stream files.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Splitting option	Whether to split the spooled file and, if so, how. Options are: ***NONE** – No splitting occurs. ***PAGE** – Splitting occurs every so many pages. ***POS** – Splitting occurs based on positional information defined in one or more CS_SPP01 structures. ***KEY** – Splitting occurs based on key information defined in one or more CS_SPK01 structures. ***POSKEY** – Splitting occurs based on positional and key information defined in one or more CS_SPP01 and CS_SPK01 structures. ***PAGGRP** – Splitting occurs based on the page groups defined in the spooled file (DDS STRPAGGRP/ENDPAGGRP keywords).	CHAR(10)
10	000A	Number of pages	The number of pages after which a new stream file is created. Used in conjunction with the value *PAGE for the previous field. Must be greater than zero if the	BINARY(4)

			previous field is *PAGE. Must be zero if the previous field is not *PAGE.	
14	000E	Split method	The method of splitting. One of the following special values must be specified. ***BEFORE*** – Splitting occurs at the end of the page preceding the page on which the split point is located. This is typically used where the split point is in a heading at the start of a group of pages. ***AFTER*** – Splitting occurs at the end of the page on which the split point is located. This is typically used where the split point is in a footing or trailer at the end of a group of pages.	CHAR(10)
24	0018	Separator character	The character that CoolSpools should insert between the main part of the file name and the numeric suffix added to the file name to create different file names for each stream file. Specify the character to use or the following special value: ***x'00'*** – No separator character should be used and the numeric suffix will follow immediately after the main body of the file name.	CHAR(1)

Structure CS_STM01 – Stream File information

Name	Description
CS_STM01	Stream file name, Allows an exit program to override the name of the stream to be created and the stream file option.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

This structure is only valid when added to the option list at the start of processing, or at the *SPLFSTR and *STMFSTR exit points, as changing the stream file name at other exit points would occur too late in the processing logic.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Stream file option	The stream file option determining how data is written to the output file. The options are: ***NONE** - The command reports an error and the existing file is not changed. For safety's sake, this is the default value. ***REPLACE** - The existing file is replaced. ***ADD** - The contents of the spooled file are appended to the end of the existing file. Please note that this option is not allowed with certain output types where data cannot be appended to without making the file unusable. ***UNIQUE** - CoolSpools generates a unique file name for the output file or files by appending a numeric suffix to the name specified on the TOSTMF parameter (before any extension). The numeric suffix will be one higher than the highest suffix associated with any existing file of this name in the directory.	CHAR(10)

			EXITPGM – The stream file option will be specified by an exit program. This option is only valid if at least one suitable exit program has been specified and only if the structure is being added at startup rather from an exit program.	
10	000A	Length of stream file name	The length of the stream file path defined in the following field.	BINARY(2)
12	000C	Stream file name	The stream file name. The length may be up to 1024 bytes long and should include the full absolute or relative path of the stream file to be output.	CHAR(*)

Structure CS_XCL01 – Excel column action

Name	Description
CS_XCL01	The CS_XCL01 structure defines actions that can be applied to Excel columns.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Column reference	An Excel column reference, e.g. A= first column, B = second column etc. up to IV.	CHAR(2)
2	0002	Column action	The action to apply to this column. Possible values are: ***MRGLFT** – Merge this column with the column to the left. Can be used to remove an unwanted column. ***MRGRGT** – Merge this column with the column to the right. Can be used to remove an unwanted column. ***ALNLFT** – Align the column contents to the left. Can be used to override the alignment applied by CoolSpools. ***ALNRGT** – Align the column contents to the right. Can be used to override the alignment applied by CoolSpools. ***DROP** – Drop the column completely. The column contents are not included in the spreadsheet. ***CVTLBL** – Convert the column to a label. Can be used to convert a numeric cell to a label.	CHAR(10)

Structure CS_XLK01 – Excluded lines by key

Name	Description
CS_XLK01	The CS_XLK01 structure defines a line or range of lines to be excluded from the output based on the appearance of a key string.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of key string	The length of the key string defined in the following field	BINARY(2)
2	0002	Key string	The key string to check for	CHAR(512)
514	0202	Number of lines	The number of lines to exclude	BINARY(4)
516	0206	From page	The first page from which the lines are to be dropped. This page number refers to the relative page number within the group of pages selected by splitting, not the absolute page number in the original spooled file. For example, if a 30-page spooled file is split into 3 10-page sections, specifying a page number of 2 on this element would refer to pages 2, 12 and 22 in the original spooled file. If this field is not supplied, it defaults to 1.	BINARY(4)
520	020A	To page	The last page from which the lines are to be dropped. This page number refers to the relative page number within the group of pages selected by splitting, not the absolute page	BINARY(4)

			number in the original spooled file. For example, if a 30-page spooled file is split into 3 10-page sections, specifying a page number of 2 on this element would refer to pages 2, 12 and 22 in the original spooled file. If this field is not supplied, it defaults to the last page in the spooled file section.	

Structure CS_XLN01 – Excluded line numbers

Name	Description
CS_XLN01	The CS_XLN01 structure defines a line or range of lines to be excluded from the output.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Line number	The first line number to be excluded	BINARY(4)
4	0004	Number of lines	The number of lines to exclude starting from the line number specified above.	BINARY(4)
8	0008	From page	The first page from which the lines are to be dropped. This page number refers to the relative page number within the group of pages selected by splitting, not the absolute page number in the original spooled file. For example, if a 30-page spooled file is split into 3 10-page sections, specifying a page number of 2 on this element would refer to pages 2, 12 and 22 in the original spooled file. If this field is not supplied, it defaults to 1.	BINARY(4)
12	000C	To page	The last page from which the lines are to be dropped. This page number refers to the relative page number within the group of pages selected by splitting, not the absolute page	BINARY(4)

			number in the original spooled file. For example, if a 30-page spooled file is split into 3 10-page sections, specifying a page number of 2 on this element would refer to pages 2, 12 and 22 in the original spooled file. If this field is not supplied, it defaults to the last page in the spooled file section.	

Structure CS_XLS01 – Excel options

Name	Description
CS_XLS01	Excel-related options.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
1	Yes	Yes	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Method	The method CoolSpools should use to identify columns in the spooled file. ***NEW** – The new method introduced with Version 6, which uses left- and right-aligned vertical “edges” in the data. ***OLD** – The old method uses by Version 4 and earlier versions, which uses the structure of the data in the spooled file and/or delimiters.	CHAR(10)
10	000A	Version	The Excel file format (BIFF version) to output. ***BIFF8** – BIFF 8 format compatible with Excel 97 and above. ***BIFF5** – BIFF 5 format compatible with Excel 5 and above.	CHAR(10)
20	0014	Page headings	How to handle page headings. ***FIRST** – Keep the first occurrence of a page heading but drop all others. ***ALL** – Keep all occurrences of a page heading. ***NONE** – Drop all occurrences of a page heading.	CHAR(10)

30	001E	Column headings	How to handle column headings. ***FIRST** – Keep the first occurrence of a column heading but drop all others. ***ALL** – Keep all occurrences of a column heading. ***NONE** – Drop all occurrences of a column heading.	CHAR(10)
40	0028	Spooled file currency symbol	The currency symbol used in the spooled file. This enables CoolSpools to locate and properly convert currency data in the spooled file. Specify the currency symbol used or the following special value: ***x'01'** – The currency symbol defined by the QCURSYM system value	CHAR(1)
41	0029	Spooled file decimal point	The decimal point used in the spooled file. This enables CoolSpools to locate and properly convert decimal data in the spooled file. Specify the decimal point symbol used or one of the following special values: ***x'01'** – The decimal point symbol defined by the QDECFMT system value ***x'02'** – The decimal point symbol defined by the DECFMT job attribute.	CHAR(1)
42	002A	Spooled file thousands separator	The thousands separator used in the spooled file. This enables CoolSpools to locate and properly convert decimal data in the spooled file. Specify the thousands separator used or one of the following special values: ***x'01'** – The thousands separator defined by the QDECFMT system	CHAR(1)

			value ***x'02'*** – The thousands separator defined by the DECFMT job attribute.	
43	002B	Spooled file date format	The date format used in the spooled file. This enables CoolSpools to locate and properly convert dates in the spooled file. Specify one of the following special values: ****SYSVAL*** – The date format defined by the QDATFMT system value ****JOB*** – The date format defined by the DATFMT job attribute. ****DMY*** – Day-Month-Year format ****MDY*** – Month-Day-Year format ****YMD*** – Year- Month-Day format	CHAR(10)
53	0035	Spooled file date separator	The date separator used in the spooled file. This enables CoolSpools to locate and properly convert dates in the spooled file. Specify the date separator character used or one of the following special values: ***x'01'*** – The date separator defined by the QDATSEP system value ***x'02'*** – The date separator defined by the DATSEP job attribute.	CHAR(1)
54	0036	Word for "page"	The word for "Page" used in the spooled file. This helps CoolSpools locate page numbers in the spooled file and identify page headings and footings. Specify the word for "Page" used in the spooled file or the following special values ****DFT*** – CoolSpools will retrieve	CHAR(20)

			the word for “Page” from the text of message id CVT0008 in message file CP_MSGF.	
74	004A	Excel date format	The format in which dates will appear in the Excel file. This is largely determined by your Excel and PC regional settings, but CoolSpools with either a 2-digit numeric month or a 3-character month abbreviation. Specify one of the following options. ***MM** – A 2-digit numeric month. ***MMM** – A 3-character alphanumeric month.	CHAR(10)
84	0054	Sheet name	The name of the worksheet in the workbook. Specify the name you want, or the following special value: ***DFT** – CoolSpools will retrieve the default sheet name from the text of message id CVT0021 in message file CP_MSGF.	CHAR(31)
115	0073	Title	The title attribute of the Excel file as displayed in the file properties.	CHAR(32)
147	0093	Subject	The subject attribute of the Excel file as displayed in the file properties.	CHAR(32)
179	00B3	Author		
211	00D3	Manager	The manager attribute of the Excel file as displayed in the file properties.	CHAR(32)
243	00F3	Company	The company attribute of the Excel file as displayed in the file properties.	CHAR(32)
275	0113	Category	The category attribute of the Excel file as displayed in the file properties.	CHAR(32)
307	0133	Keywords	The keywords attribute of the	CHAR(128)

			Excel file as displayed in the file properties.	
435	01B3	Comments	The comment attribute of the Excel file as displayed in the file properties.	CHAR(256)

Structure CS_XPK01 – Excluded pages by key

Name	Description
CS_XPK01	The CS_XPK01 structure defines a page or range of pages to be excluded from the output based on the appearance of a key string.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Length of key string	The length of the key string defined in the following field	BINARY(2)
2	0002	Key string	The key string to check for	CHAR(512)
514	0202	Option	Whether to exclude pages that contain the key string or pages that do not contain the key string. Possible values are: ***CT*** – Exclude pages that contain the key string. ***NC*** – Exclude pages that do not contain the key string.	CHAR(10)
524	020C	Number of pages	The number of pages to exclude	BINARY(4)

Structure CS_XPN01 – Excluded page numbers

Name	Description
CS_XLP01	The CS_XPN01 structure defines a page or range of pages to be excluded from the output.

Maximum	*SPLFSTR	*STMFSTR	*PAGESTR	*SPLFEND	*STMFEND	*PAGEEND	*PAGECTL
100	No	No	No	No	No	No	No

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Position at which pages should be excluded	The position at which pages should be excluded. Possible values are: ***SPLFSTR*** – At the start of the spooled file. ***STMFSTR*** – At the start of the stream file ***SPLFEND*** – At the end of the spooled file. ***STMFEND*** – At the end of the stream file ***PAGNBR*** – At a specified page number. The page number to be excluded is defined in the following field.	CHAR(10)
10	000A	Number of pages or page number	The number of pages to exclude at the point specified above, or, if the previous field is *PAGNBR, the number of the page in the original spooled file to be excluded. The following special value is allowed if the previous field is not *PAGNBR: ***-1*** – Exclude blank pages at the specified position.	BINARY(4)

APIs for managing the option structure list

CoolSpools provides a set of APIs for managing the option structure list.

Although it is possible to manage the option structure list yourself, you are strongly recommended to use the APIs provided, which will not only make the job a lot simpler, but also provide protection against any future changes to the format of the list.

Please note the following general points:

1. The APIs are available in two forms:
    - A set of bindable procedures exported from CoolSpools service program CS_SRVPGM. You will need to bind service program CS_SRVPGM to your exit program. You will also need to bind in service program CP_SRVPGM as that contains some of the run-time variables the option list APIs modify. To bind in these service programs, specify

      CRTPGM …

      BNDSRVPGM(COOLSPV7R1/CP_SRVPGM

      COOLSPV7R1/CS_SRVPGM)
    - A set of callable program objects which act as wrappers to the bindable APIs.
2. If you are calling these APIs (either the bindable or callable forms) in a program to prepare the option structure list before calling the Conversion API, it is recommended that this program runs in a new activation group, i.e. ACTGRP(*NEW) should be specified on the CRTPGM command.
3. If you are calling these APIs (either the bindable or callable forms) from a CoolSpools exit program, the exit program ***must*** be defined to run in the caller's activation group, i.e. **ACTGRP(*CALLER)** should be specified on the CRTPGM command. If this is not done, the exit program will not be able to update the option structure list because the storage it occupies will exist in a different activation group. For this reason, exit programs must be written in an ILE language if you wish to call the option list APIs to modify CoolSpools options at run time, since OPM programs cannot run in the same activation group as CoolSpools.
4. The first API to be called should always be **OptInitialize.** This need be called only once (though no harm will be done if it is called multiple times). You can determine whether **Optinitialize** has already been called by checking the Boolean flag **xlOptInitialized** which is also exported from service program CS_SRVPGM.
5. The last API to be called should always be **OptTerminate.** Calling OptTerminate releases resources and flagged the option structure environment as un-initialized.
6. All bindable option structure list APIs (except OptRtvError) return an integer return code indicating the success or failure of the operation. The equivalent

callable APIs (except AR_RTVERRR) have this return code as their first parameter.

7. The possible return codes are:
    - < 0 = error occurred
    - 0 = operation completed normally
    - > 0 = operation completed normally, but no data was returned (e.g. end-of-list condition or structure not found).
8. If the return code is less than zero, additional information about the error condition can be obtained by calling the OptRtvError (bindable) or AR_RTVERR (callable) function.
9. Note that for the callable APIs, the first parameter is always the return code and this is followed immediately by the other parameters in the same form and sequence as the parameters to the equivalent bindable API.

Before processing a page of data, CoolSpools will assess which option structures are in effect and should influence its operations. The selection of option structures is governed by the following considerations.

1. When adding a structure to the option structure list, it can be assigned a *page scope*. If a page scope is assigned, the options selected by the option structure will apply only to the page or pages indicated by the page scope. If no page scope is assigned, the option structure is assumed to relate to all pages (known as “global page scope”).
2. When adding a structure to the option structure list, it must be assigned a *priority*, which can be either:
    - Primary. A structure with primary priority will always be selected so long as it is scoped to the range of pages being processed.
    - Secondary. A structure with secondary page scope will only be selected if it has appropriate page scope and no other structure of the same format with primary priority has been selected. Secondary priority thereby provides a means of indicating default values.
3. Options defined on CoolSpools command parameters are converted into option structures with secondary priority. For example, a password provided on the PASSWORD parameter of the CVTSPLPDF command will be added to the option structure list in a CS_PWD01 structure with secondary priority. This means that the password may be overridden by an exit program which provides a CS_PWD01 structure with primary priority. If, however, the exit program does not provide such a CS_PWD01 structure for a particular stream file, the password specified on the PASSWORD parameter of the CVTSPLPDF command will be used by default.

Constants and prototypes needs to call the option list APIs are provided in copybook AR_OPTFNCP.

OptInitialize

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details)	BINARY(4)

Required parameter group
None

Callable equivalent: AR_INZOPTR

The OptInitialize function sets up the option structure environment. It flags the environment as initialized by setting flag xlOptInitialized to TRUE (‘1’).

OptInitialize should always be called before calling any other option structure API.

If you call other option structure APIs before calling OptInitialize, an error will occur.

If you are calling option structure APIs from within CoolSpools exit programs, CoolSpools will already have called OptInitialize itself before calling the exit program, so calling OptInitialize again should not be necessary.

If, however, you are calling option structure APIs in order to set up the option structure list before calling the CoolSpools Spool Conversion API, you must ensure that OptInitialize is called first.

Since the option structure list APIs are scoped to a single activation group, OptInitialize needs to be called for each activation group.

OptTerminate

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details)	BINARY(4)

Required parameter group
None

Callable equivalent: AR_ENDOPTR

The OptTerminate function releases resources and deletes the option structure environment. It flags the environment as un-initialized by setting flag xlOptInitialized to FALSE (‘0’).

OptTerminate should always be the last option structure list API called during a conversion run.

OptCrtList

The OptCrtList function creates an option structure list for use with CoolSpools.

Before you can add option structures to the structure list, you must first create it.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details)	BINARY(4)

Required parameter group
None

Callable equivalent: AR_CRTLSTR

OptDltList

The OptDltList function deletes the current option structure.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details)	BINARY(4)

Required parameter group
None

Callable equivalent: AR_ENDOPTR

OptInzList

The OptInzList function creates a new option list structure and populates it with the data supplied to it.

The primary use of this API is to create an option structure list in one activation group from an option structure list prepared in a different activation group. For example an application might prepare an option structure list for passing to the CoolSpools Spool Conversion API by calling the option structure list APIs. The option structure list will then be passed to the CoolSpools Spool Conversion API on ninth parameter. Since CoolSpools runs in its own activation group it will then create a new option structure list from the data provided using the OptInzList API.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details)	BINARY(4)

Required parameter group			
1	Length of option structure list data	Input	BINARY(4)
2	Option structure list data	Input	CHAR(*)

Callable equivalent: AR_INZLSTR

Required Parameter Group

Length of option structure list data

INPUT; BINARY(4)

The length of the data supplied on the following parameter.

The **OptRtvLength** API can be used to determine the total length of the option structure list.

Option structure list data

INPUT; CHAR(*)

The data must be properly structured option structure list and should be prepared by calling the option structure list APIs.

The OptRtvList or OptRtvData API can be used to access or retrieve the data for passing to OptInzList.

OptRtvList

The OptRtvList function retrieves a pointer to the option structure list and the length of the option structure list so the list can be passed to the CoolSpools Spool Conversion API.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details)	BINARY(4)

Required parameter group			
1	Length of option structure list data	Output	BINARY(4)
2	Pointer to option structure list	Output	POINTER

Callable equivalent: None (Pointer parameter not supported). Use AR_RTVDTAR instead.

Required Parameter Group

Length of option structure list data

OUTPUT; BINARY(4)

The total length of the data in the option structure list is returned in this field.

Pointer to option structure list data

OUTPUT; POINTER

A pointer to the start of the option structure list is returned in this field.

You can use this pointer as a basing pointer for a structure and then pass the structure to the CoolSpools Spool Conversion API. This avoids having to return all of the option structure list data to the program simply to pass it to CoolSpools.

OptRtvData

The OptRtvData function retrieves the data contents of the option structure list into a variable. The data can then be passed to CoolSpools.

OptRtvData differs from OptRtvList in that it retrieves the actual data content of the list rather than just a pointer to the list. It is provided primarily for the benefit of those who prefer to avoid the use of pointers.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details)	BINARY(4)

Required parameter group			
1	Length of the receiver variable	Input	BINARY(4)
2	Receiver variable	Output	CHAR(*)
3	Length of data returned	Output	BINARY(4)

Callable equivalent: AR_RTVDTAR

Required Parameter Group

Length of the receiver variable

INPUT; BINARY(4)

The length of the variable specified on the next parameter.

Receiver variable

OUTPUT; CHAR(*)

The variable to receive the option list data.

Please note that the option structure list can easily exceed the maximum size of an RPG variable (32 kb or 64 kb depending on the version of IBM I (OS/400) you are running). Unless you have added only a few small option structures to the list, it is likely that you will need to use a user space or allocate dynamic storage to hold the full list.

You can call the **OptRtvLength** API to determine the required size of the receiver variable and allocate sufficient storage before calling OptRtvData.

Length of data returned

INPUT; BINARY(4)

The length of the option structure list data returned in the variable specified on the previous parameter.

OptRtvLength

The OptRtvLength function retrieves the total length of the data contents of the option structure list. This information can be used to ensure that sufficient storage is available for a subsequent call to OptRtvData.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details)	BINARY(4)

Required parameter group			
1	Length of option structure list data	Output	BINARY(4)

Callable equivalent: AR_RTVLENR

Required Parameter Group

Length of option structure list data

OUTPUT; BINARY(4)

The total length of the option structure list.

OptAddItem

The OptAddItem function adds an option structure to the option structure list.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details)	BINARY(4)

Required parameter group			
1	Format of option structure	Input	CHAR(8)
2	Length of option structure data	Input	BINARY(4)
3	Option structure data	Input	CHAR(*)

Optional parameter group 1			
4	Option structure priority	Input	CHAR(1)

Optional parameter group 2			
5	Scope from page	Input	BINARY(4)
6	Scope to page	Input	BINARY(4)

Optional parameter group 3			
7	Returned option structure identifier	Output	BINARY(4)

Callable equivalent: AR_ADDITMR

Required Parameter Group

Format of option structure

INPUT; CHAR(8)

A name identifying the type of option structure which is being created. For example, PDF-related options are defined using a structure named CS_PDF01.

The name must be a predefined CoolSpools option structure name.

Option structures are defined in copybook CS_CVTAPID.

Length of option structure data

INPUT; BINARY(4)

The length of the option structure data provide on the following parameter.

Option structure data

INPUT; CHAR(*)

The option structure data to be added to the option structure list.

Optional parameter group 1

Option structure priority

INPUT; CHAR(1)

A code identifying the priority of the option structure being added to the list.

The following values are supported for this parameter:

1	Primary. So long as the page scope is appropriate, the option structure will always be selected and will influence CoolSpools processing.
2	Secondary. The option structure will only be selected and influence CoolSpools processing if the page scope is appropriate and no other structure of this format with primary priority has already been selected.

If this parameter is not passed, secondary priority is assumed.

Optional parameter group 2

Scope from page

INPUT; BINARY(4)

The first page to which this structure relates.

Scope to page

INPUT; BINARY(4)

The last page to which this structure relates.

Optional parameter group 3

Returned option structure identifier

OUTPUT; BINARY(4)

An identifier which uniquely identifies the option structure just added. This identifier may be used on subsequent calls to **OptRtvItem**, **OptUpdItem** and **OptRmvItem**.

OptRtvItem

The OptRtvItem function retrieves the details of an option structure from the option structure list.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details) > 1 = No structure returned. Either the beginning or end of the list was reached or the option structure identifier specified does not exist.	BINARY(4)

Required parameter group			
1	Option structure format to retrieve	Input	CHAR(8)
2	Option structure to retrieve	Input	BINARY(4)
3	Identifier of option structure returned	Output	BINARY(4)
4	Format of option structure returned	Output	CHAR(8)
5	Priority of option structure returned	Output	CHAR(1)
6	Scope from-page of option structure returned	Output	BINARY(4)
7	Scope to-page of option structure returned	Output	BINARY(4)
8	Context of option structure returned	Output	CHAR(10)
9	Offset of option structure returned	Output	BINARY(4)
10	Length of data buffer	Input	BINARY(4)
11	Data buffer	Output	CHAR(*)
12	Length of data returned	Output	BINARY(4)
13	Length of data available	Output	BINARY(4)

Callable equivalent: AR_RTVITMR

Required Parameter Group

Option structure format to retrieve

INPUT; CHAR(8)

The format name of the type of structure to retrieve.

The name must be a predefined CoolSpools option structure name or the following special value:

**ANY*	CoolSpools will retrieve option structures irrespective of their format name.

Option structure to retrieve

INPUT; BINARY(4)

This parameter identifies which option structure should be retrieved. It can be either an option structure identifier returned by **OptAddItem** to select a specific structure added earlier or one of the following special values:

-1	CoolSpools will retrieve the first option structure in the list which matches the format name specified on the previous parameter.
-2	CoolSpools will retrieve the last option structure in the list which matches the format name specified on the previous parameter.
-3	CoolSpools will retrieve the next option structure in the list which matches the format name specified on the previous parameter.
-4	CoolSpools will retrieve the previous option structure in the list which matches the format name specified on the previous parameter.

Processing starts with the current option structure, i.e. the most recently retrieved option structure. If there is no current option structure, an error will occur. You should always position to an option structure in the list using either -1 (first) or -2 (last) before using -3 (next) or -4 (previous).

Identifier of option structure returned

OUTPUT; BINARY(4)

The unique identifier of the option structure returned.

Format of option structure returned

OUTPUT; CHAR(8)

The format name of the option structure returned.

Priority of option structure returned

OUTPUT; CHAR(1)

The priority of the option structure returned.

1	Primary. So long as the page scope is appropriate, the option structure will always be selected and will influence CoolSpools processing.
2	Secondary. The option structure will only be selected and influence CoolSpools processing if the page scope is appropriate and no other structure of this format with primary priority has already been selected.
3	Tertiary. The option structure is a system-generated default and will only be selected and influence CoolSpools processing if the page scope is appropriate and no other structure of this format with primary priority has already been selected.

Scope from-page of option structure returned

OUTPUT; BINARY(4)

The first page to which the returned structure relates.

Scope to-page of option structure returned

OUTPUT; BINARY(4)

The last page to which the returned structure relates.

Context of option structure returned

INPUT; CHAR(10)

The context of the returned option structure. The context indicates the stage of processing at which the option structure was added.

**STRUP*	Startup. The structure was added before the CoolSpools Spool Conversion API was called.
**SYSDFT*	System default. The structure was system-generated because no default value had been provided.
**SPLFSTR	The structure was added by an exit program called at the *SPLFSTR exit point.
**STMFSTR	The structure was added by an exit program called at the *STMFSTR exit point.
**PAGESTR	The structure was added by an exit program called at the *PAGESTR exit point.

*SPLFEND The structure was added by an exit program called at the *SPLFEND exit point.

*STMFEND The structure was added by an exit program called at the *STMFEND exit point.

*PAGEEND The structure was added by an exit program called at the *PAGEEND exit point.

*PAGECTL The structure was added by an exit program called at the *PAGECTL exit point.

Offset of option structure returned

OUTPUT; BINARY(4)

The offset from the start of the option structure list to the start of the data for this structure.

Using this API, you can retrieve the associated with the option structure in the buffer specified in the next two parameters. However, if preferred, you calculate a basing pointer to access the data using this offset value and the pointer to the start of the option structure list returned with OptRtvList. If you do this, you must ensure that the data is not corrupted by writing to the option structure list without using the APIs provided.

Length of data buffer

INPUT; BINARY(4)

The length of the data buffer provided on the following parameter.

Data buffer

OUTPUT; CHAR(*)

The data buffer in which the option structure data is returned.

Length of data returned

OUTPUT; BINARY(4)

The length of the data returned in the data buffer.

Length of data available

OUTPUT; BINARY(4)

The length of the data available for the selected option structure in the option structure list. This may be longer than the length of the data returned if the data buffer provided was too small to receive all of the data.

OptUpdItem

The OptUpdItem function updates an option structure in the option structure list.

Please note that the option structure is not in fact updated. In reality, the existing structure is deleted and a new structure added with the same option identifier.

It is unusual for the OptUpdItem function to be necessary. Where options need to change from one stream file to the next or from one page to the next, it is usually simpler just to use OptAddItem to add a new option structure to the list with appropriate page scope rather than going to the trouble of retrieving an existing option structure and calling OptUpdItem.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details) > 1 = Structure not found.	BINARY(4)

Required parameter group			
1	Option identifier to update	Input	BINARY(4)
2	Format of option structure to update	Input	CHAR(8)
3	Length of option structure data	Input	BINARY(4)
4	Option structure data	Input	CHAR(*)

Optional parameter group 1			
5	Option structure priority	Input	CHAR(1)

Optional parameter group 2			
6	Scope from page	Input	BINARY(4)
7	Scope to page	Input	BINARY(4)

Callable equivalent: AR_UPDITMR

Required Parameter Group

Option identifier to update

INPUT; BINARY(4)

The unique option structure identifier of the option structure to be updated. This can have been returned by the **OptAddItem** or **OptRtvItem** APIs.

Format of option structure

INPUT; CHAR(8)

A name identifying the type of option structure which is to be written to the option structure list. For example, PDF-related options are defined using a structure named CS_PDF01.

It is possible to replace an option structure of one format with an option structure of a different format, although it would be unusual for this to be necessary.

The name must be a predefined CoolSpools option structure name.

Option structures are defined in copybook CS_CVTAPID.

Length of option structure data

INPUT; BINARY(4)

The length of the option structure data provided on the following parameter.

Option structure data

INPUT; CHAR(*)

The option structure data to be written to the option structure list replacing the existing option structure data.

Optional parameter group 1

Option structure priority

INPUT; CHAR(1)

A code identifying the priority of the option structure being added to the list.

The following values are supported for this parameter:

1	Primary. So long as the page scope is appropriate, the option structure will always be selected and will influence CoolSpools processing.
2	Secondary. The option structure will only be selected and influence CoolSpools processing if the page scope is appropriate and no other structure of this format with primary priority has already been selected.

If this parameter is not passed, secondary priority is assumed.

Optional parameter group 2

Scope from page

INPUT; BINARY(4)

The first page to which this structure relates.

Scope to page

INPUT; BINARY(4)

The last page to which this structure relates.

OptChkItem

The **OptChkItem** function checks for the existence of an option structure in the option structure list.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details) > 1 = Structure not found.	BINARY(4)

Required parameter group			
1	Format of option structure to check for	Input	CHAR(8)
2	Found/not found	Output	CHAR(1)

Optional parameter group 2			
3	Minimum priority	Input	CHAR(1)

Callable equivalent: AR_CHKITMR

Required Parameter Group

Format of option structure to check for

INPUT; CHAR(8)

The unique option structure identifier of the option structure to be located in the option list.

Found/not found

OUTPUT; CHAR(1)

A boolean flag returned to indicate whether the specified option exists in the option list.

The value returned will be one of the following:

1	An option of this type exists in the option list.
0	No option of this type exists in the option list

Optional parameter group 1

Minimum priority

INPUT; CHAR(1)

The minimum priority level to check for. Options with priorities higher than the specified level will be ignored.

OptRmvItem

The **OptRmvItem** function removes an option structure from the option structure list.

It is unusual for the OptRmvItem function to be necessary. Where options need to change from one stream file to the next or from one page to the next, it is usually simpler just to use page scope to control which pages an option structure relates to.

Return value	
Return code 0 = OK < 0 = Error occurred (call OptRtvError for details) > 1 = Structure not found.	BINARY(4)

Required parameter group			
1	Option identifier to remove	Input	BINARY(4)

Callable equivalent: AR_RMVITMR

Required Parameter Group

Option identifier to remove

INPUT; BINARY(4)

The unique option structure identifier of the option structure to be updated. This can have been returned by the **OptAddItem** or **OptRtvItem** APIs.

OptRtvError

The OptRtvError function returns details of the most recent error.

Return value	
None	

Required parameter group			
1	Error message identifier	Output	CHAR(7)
2	Size of substitution data buffer	Input	BINARY(4)
3	Substitution data buffer.	Output	CHAR(*)
4	Length of substitution data returned.	Output	BINARY(4)

Callable equivalent: AR_RTVERRR

Required Parameter Group

Error message identifier

OUTPUT; CHAR(7)

The message identifier that identifies the type of error that occurred.

CoolSpools messages start with the prefix CVT and are located in message file CP_MSGF.

Generic ariadne messages start with the prefix ARI and are located in message file AR_MSGF.

Size of substitution data buffer

INPUT; BINARY(4)

The size of the buffer provided for the substitution data (next parameter).

Substitution data buffer

OUTPUT; BINARY(4)

The available substitution data for the message is returned in this buffer, up to the length specified on the preceding field or a maximum of 1,024 bytes.

Length of substitution data returned

INPUT; BINARY(4)

The length of the data returned in the previous field.

Examples

The following example program demonstrates how to call the option list APIs to create an option structure list and add option structures to the list. It then shows how to call the CoolSpools Spool Conversion API and interpret the data returned by it.

The source of this example program is supplied in CoolSpools source file CS_SRCFILE.

If should be created as follows.

1. Create the module:

```
CRTRPGMOD MODULE(CVTEXAMPLE) SRCFILE(CS_SRCFILE)
```

2. Create the program. Service program CS_SRVPGM must be bound in so as to make available the option list APIs.

```
CRTPGM PGM(CVTEXAMPLE) BNDSRVPGM(CS_SRVPGM)
```

```
 ****************************************************************
 *                                                              *
 * PROGRAM NAME: CVTEXAMPLE                                     *
 *                                                              *
 * DESCRIPTION : Demonstrates how to call the CoolSpools        *
 *               Conversion API and option list APIs from ILE   *
 *               RPG.                                           *
 *                                                              *
 ****************************************************************

H EXTBININT

 * Copybook for option list structure types
 /COPY CS_SRCFILE,CS_CvtAPID

 * Copybook for conversion API prototype
 /COPY CS_SRCFILE,CS_CvtAPIP

 * Copybook for option list API prototypes
 /COPY CS_SRCFILE,AR_OPTFNCP

 * Prototype for API to send a program message
D SndPgmMsg       PR                  EXTPGM('QMHSNDPM')
D  iaMsgId                       7A   CONST
D  iaMsgFile                    20A   CONST
D  iaMsgDta                  32767A   CONST OPTIONS(*VARSIZE)
D  iiMsgDtaLen                  10I 0 CONST
D  iaMsgType                    10A   CONST
D  iaStackEntry                 10A   CONST
D  iiStackCount                 10I 0 CONST
D  oaMsgKey                      4A
D  baAPIError                 1024A         OPTIONS(*VARSIZE)

 * IBM API error structure
D APIError        DS
D  siAPIErrSize                 10I 0 INZ(%size(APIError))
D  siAPIErrLen                  10I 0 INZ(0)
D  saAPIErrMsg                   7A
D                                1A
D  saAPIErrDta                 256A

 * General error flag
D wlError         S                N

 * Flag indicating an error on a call to an option list API
```

```
D wlOptAPIError   S               N

 * Return code
D wiRtnCode       S             10I 0

 * Length of option structure list
D wuOptionDta     S             10U 0

 * Basing pointer for option structure list data
D wpOptionDta     S               *

 * Option structure list data
D waOptionDta     S          32767A   BASED(wpOptionDta)

 * Error message identifying the type of error that occurred
D waMsgId         S              7A

 * Associated error message data
D waMsgDta        S           1024A

 * Length of associated error message data
D wuMsgDtaLen     S             10U 0

 * Qualified message file name
D waMsgFile       S             20A

 * Message key returned from SNDPGMMSG API
D waMsgKey        S              4A

 ******************************************************************
 * Set up the error structure
 ******************************************************************
C                   CLEAR                   CS_ERR01
C                   EVAL      suErr01Size = %size(CS_ERR01)

 ******************************************************************
 * Set up the return structure
 ******************************************************************
C                   CLEAR                   CS_RTN01
C                   EVAL      suRtn01Size = %size(CS_RTN01)

 ******************************************************************
 * Call the initialization routine for the option structure list APIs
 ******************************************************************
C                   EVAL      wiRtnCode = OptInitialize
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF

 ******************************************************************
 * Create the option structure list
 ******************************************************************
C                   IF        not wlOptAPIError
C                   EVAL      wiRtnCode = OptCrtList
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ******************************************************************
 * Add option structure for email parameters
 ******************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                   CS_EML01
C                   EVAL      saEml01Option = '*YES'
C                   EVAL      saEml01Delete = '*NO'
C                   EVAL      svEml01Subjct = 'Example email'
```

```
C                   EVAL      saEml01Method = '*ATTACH'
C                   EVAL      saEml01Prty = 'N'
C                   EVAL      saEml01Confrm = 'N'
C                   EVAL      saEml01SndMlt = 'N'
C                   EVAL      svEml01FrmEml = 'demo@ariadnesoftware.co.uk'
C                   EVAL      svEml01FrmNam = 'Demo'
C                   EVAL      svEml01MsgTxt = 'API Demo'
C                   EVAL      saEml01Format = '*BOTH'
C                   EVAL      wiRtnCode = OptAddItem('CS_EML01'     :
C                                                    %size(CS_EML01):
C                                                    CS_EML01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ********************************************************************
 * Add option structure for email recipient
 ********************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                   CS_EMT01
C                   EVAL      svEMT01Email = '*EXITPGM'
C                   EVAL      svEMT01Name = '*EXITPGM'
C                   EVAL      saEMT01Type = '*EXITPGM'
C                   EVAL      wiRtnCode = OptAddItem('CS_EMT01'     :
C                                                    %size(CS_EMT01):
C                                                    CS_EMT01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON

C                   ENDIF
C                   ENDIF

 ********************************************************************
 * Add option structure for splitting
 ********************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                   CS_SPT01
C                   EVAL      saSPT01Option = '*POS'
C                   EVAL      saSPT01Method = '*BEFORE'
C                   EVAL      saSPT01SepChr = x'00'
C                   EVAL      wiRtnCode = OptAddItem('CS_SPT01'     :
C                                                    %size(CS_SPT01):
C                                                    CS_SPT01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ********************************************************************
 * Add option structure for positional splitting
 ********************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                   CS_SPP01
C                   EVAL      snSPP01Y = 5
C                   EVAL      snSPP01X = 9
C                   EVAL      snSPP01Length = 3
C                   EVAL      saSPP01Unit = '*ROWCOL'
C                   EVAL      saSPP01Oper = '*NE'
C                   EVAL      svSPP01CmpStr = '*PRV'
C                   EVAL      wiRtnCode = OptAddItem('CS_SPP01'     :
C                                                    %size(CS_SPP01):
C                                                    CS_SPP01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF
```

```
 ****************************************************************
 * Add option structures for positional exit program parameters
 ****************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                 CS_EPP01
C                   EVAL      siEPP01PagNbr = 1
C                   EVAL      snEPP01Y = 5
C                   EVAL      snEPP01X = 9
C                   EVAL      snEPP01Length = 3
C                   EVAL      saEPP01Unit = '*ROWCOL'
C                   EVAL      wiRtnCode = OptAddItem('CS_EPP01'     :
C                                                    %size(CS_EPP01):
C                                                    CS_EPP01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

C                   IF        not wlOptAPIError
C                   CLEAR                 CS_EPP01
C                   EVAL      siEPP01PagNbr = 1
C                   EVAL      snEPP01Y = 6
C                   EVAL      snEPP01X = 9
C                   EVAL      snEPP01Length = 5
C                   EVAL      saEPP01Unit = '*ROWCOL'
C                   EVAL      wiRtnCode = OptAddItem('CS_EPP01'     :
C                                                    %size(CS_EPP01):
C                                                    CS_EPP01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

C                   IF        not wlOptAPIError
C                   CLEAR                 CS_EPP01
C                   EVAL      siEPP01PagNbr = 1
C                   EVAL      snEPP01Y = 6
C                   EVAL      snEPP01X = 9
C                   EVAL      snEPP01Length = 30
C                   EVAL      saEPP01Unit = '*ROWCOL'
C                   EVAL      wiRtnCode = OptAddItem('CS_EPP01'     :
C                                                    %size(CS_EPP01):
C                                                    CS_EPP01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ****************************************************************
 * Add option structure for exit program to supply email recipient
 * and stream file name and set the passwords.
 ****************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                 CS_EXT01
C                   EVAL      saExt01Pgm = 'EXITTYPE3'
C                   EVAL      saExt01Lib = '*LIBL'
C                   EVAL      saExt01Type = '*TYPE3'
C                   EVAL      saExt01When = '*STMFSTR'
C                   EVAL      wiRtnCode = OptAddItem('CS_EXT01'     :
C                                                    %size(CS_EXT01):
C                                                    CS_EXT01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF
```

```
 ****************************************************************
 * Retrieve a pointer to the option structure list and its length
 ****************************************************************
C                   IF        not wlOptAPIError
C                   EVAL      wiRtnCode = OptRtvList(wuOptionDta:
C                                                    wpOptionDta)
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ****************************************************************
 * Call the CoolSpools Spool Conversion API
 ****************************************************************
C                   IF        not wlOptAPIError
C                   CALLP     CS_CvtAPI('QSYSPRT ' :
C                                       '*'        :
C                                       -1         :
C                                       8          :
C                                       '*EXITPGM' :
C                                       '*PDF'     :
C                                       '*EXITPGM' :
C                                       CS_ERR01   :
C                                       wuOptionDta:
C                                       waOptionDta:
C                                       CS_RTN01   )
C                   ENDIF

 ****************************************************************
 * Delete the option structure list
 ****************************************************************
C                   IF        not wlOptAPIError
C                   EVAL      wiRtnCode = OptDltList
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

C                   SELECT

 ****************************************************************
 * If an error occurred calling the conversion API, get the error
 * details from the error structure. The available length returned
 * includes the length of the error message id and the following one-
 * byte reserved area (for consistency with IBM APIs).
 ****************************************************************
C                   WHEN      suErr01Avail > 0
C                   EVAL      wlError = *ON
C                   EVAL      waMsgId = saErr01MsgId
C                   EVAL      wuMsgDtaLen = suErr01Avail
C                   IF        suErr01Avail > %size(saErr01MsgId) + 1
C                   EVAL      wuMsgDtaLen = suErr01Avail
C                                         - %size(saErr01MsgId) - 1
C                   IF        wuMsgDtaLen > %size(waMsgDta)
C                   EVAL      wuMsgDtaLen = %size(waMsgDta)
C                   ENDIF
C                   EVAL      waMsgDta = %subst(saErr01MsgDta:
C                                               1            :
C                                               wuMsgDtaLen  )
C                   ELSE
C                   EVAL      wuMsgDtaLen = 0
C                   ENDIF

 ****************************************************************
 * If an error occurred calling an option structure list API, get the
 * error details
 ****************************************************************
```

```
C                   WHEN      wlOptAPIError
C                   EVAL      wlError = *ON
C                   CALLP     OptRtvError(waMsgId        :
C                                         %size(waMsgDta):
C                                         waMsgDta       :
C                                         wuMsgDtaLen    )

C                   ENDSL

 ********************************************************************
 * Close down the option structure list environment. Do this even if
 * an error occurred previously.
 ********************************************************************
C                   IF        xlOptInitialized
C                   CALLP     OptTerminate
C                   ENDIF

 ********************************************************************
 * Insert error handling appropriate to your environment here.
 * Here we just send an escape message which will abort processing.
 ********************************************************************
C                   IF        wlError

 * Set the message file name

C                   SELECT
C                   WHEN      %subst(waMsgId:1:3) = 'CVT'
C                   EVAL      waMsgFile = 'CP_MSGF   *LIBL'
C                   WHEN      %subst(waMsgId:1:3) = 'ARI'
C                   EVAL      waMsgFile = 'AR_MSGF   *LIBL'
C                   OTHER
C                   EVAL      waMsgFile = 'QCPFMSG   *LIBL'
C                   ENDSL

C                   CALLP     SndPgmMsg(waMsgId    :
C                                       waMsgFile  :
C                                       waMsgDta   :
C                                       wuMsgDtaLen:
C                                       '*ESCAPE'  :
C                                       '*PGMBDY'  :
C                                       1          :
C                                       waMsgKey   :
C                                       APIError   )
C                   ELSE

 ********************************************************************
 * Send a completion message
 ********************************************************************
C                   EVAL      waMsgId = 'CPF9898'
C                   EVAL      waMsgFile = 'QCPFMSG   *LIBL'
C                   EVAL      waMsgDta = %trim(%editc(suRtn01StmCnt:'Z'))
C                                      + ' stream files created. '
C                                      + %trim(%editc(suRtn01EmlCnt:'Z'))
C                                      + ' emails sent'
C                   EVAL      wuMsgDtaLen = %len(%trimr(waMsgDta))
C                   CALLP     SndPgmMsg(waMsgId    :
C                                       waMsgFile  :
C                                       waMsgDta   :
C                                       wuMsgDtaLen:
C                                       '*COMP'    :
C                                       '*PGMBDY'  :
C                                       1          :
C                                       waMsgKey   :
C                                       APIError   )

C                   ENDIF
```

```
C                   EVAL      *INLR = *ON
C                   RETURN
```

The following example program demonstrates how to prepare the option list APIs using the callable versions of the option list APIs.

The source of this example program is supplied in CoolSpools source file CS_SRCFILE.

The program tells CoolSpools to call the *TYPE2 example exit program EXITTYPE2 and the *TYPE1 example exit program EXITTYPE1, the source of which is also supplied in CS_SRCFILE.

If should be created as follows.

1. Create the module:

```
CRTRPGMOD MODULE(CVTEXAMPL2) SRCFILE(CS_SRCFILE)
```

2. Create the program. Service program CS_SRVPGM must be bound in so as to make available the option list APIs.

```
CRTPGM PGM(CVTEXAMPL2) BNDSRVPGM(CS_SRVPGM)
```

```
*****************************************************************
*                                                               *
* PROGRAM NAME: CVTEXAMPL2                                      *
*                                                               *
* DESCRIPTION : Demonstrates how to call the CoolSpools         *
*               Conversion API and the callable option list APIs *
*                                                               *
*****************************************************************

H EXTBININT

 * Copybook for option list structure types
 /COPY CS_SRCFILE,CS_CvtAPID

 * Copybook for conversion API prototype
 /COPY CS_SRCFILE,CS_CvtAPIP

 * Copybook for option list API prototypes
 /COPY CS_SRCFILE,AR_OPTFNCP

 * Prototype for API to send a program message
D SndPgmMsg       PR                  EXTPGM('QMHSNDPM')
D  iaMsgId                       7A   CONST
D  iaMsgFile                    20A   CONST
D  iaMsgDta                  32767A   CONST OPTIONS(*VARSIZE)
D  iiMsgDtaLen                  10I 0 CONST
D  iaMsgType                    10A   CONST
D  iaStackEntry                 10A   CONST
D  iiStackCount                 10I 0 CONST
D  oaMsgKey                      4A
D  baAPIError                 1024A         OPTIONS(*VARSIZE)

 * IBM API error structure
D APIError        DS
D  siAPIErrSize                 10I 0 INZ(%size(APIError))
D  siAPIErrLen                  10I 0 INZ(0)
D  saAPIErrMsg                   7A
D                                1A
D  saAPIErrDta                 256A

 * General error flag
```

```
D wlError         S               N

 * Flag indicating an error on a call to an option list API
D wlOptAPIError   S               N

 * Return code
D wiRtnCode       S             10I 0

 * Length of option structure list
D wuOptionDta     S             10U 0

 * Basing pointer for option structure list data
D wpOptionDta     S               *

 * Option structure list data
D waOptionDta     S          32767A   BASED(wpOptionDta)


 * Error message identifying the type of error that occurred
D waMsgId         S              7A

 * Associated error message data
D waMsgDta        S           1024A

 * Length of associated error message data
D wuMsgDtaLen     S             10U 0

 * Qualified message file name
D waMsgFile       S             20A

 * Message key returned from SNDPGMMSG API
D waMsgKey        S              4A

 ******************************************************************
 * Set up the error structure
 ******************************************************************
C                   CLEAR                   CS_ERR01
C                   EVAL      suErr01Size = %size(CS_ERR01)

 ******************************************************************
 * Set up the return structure
 ******************************************************************
C                   CLEAR                   CS_RTN01
C                   EVAL      suRtn01Size = %size(CS_RTN01)

 ******************************************************************
 * Call the initialization routine for the option structure list APIs
 ******************************************************************
C                   IF        not xlOptInitialized
C                   CALLP     AR_INZOPTR(wiRtnCode)
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ******************************************************************
 * Create the option structure list
 ******************************************************************
C                   IF        not wlOptAPIError
C                   CALLP     AR_CRTLSTR(wiRtnCode)
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ******************************************************************
 * Add option structure for splitting
```

```
 ****************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                   CS_SPT01
C                   EVAL      saSPT01Option = '*POS'
C                   EVAL      saSPT01Method = '*BEFORE'
C                   EVAL      saSPT01SepChr = x'00'
C                   EVAL      wiRtnCode = OptAddItem('CS_SPT01'     :
C                                                    %size(CS_SPT01):
C                                                    CS_SPT01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ****************************************************************
 * Add option structure for positional splitting
 ****************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                   CS_SPP01
C                   EVAL      snSPP01Y = 5
C                   EVAL      snSPP01X = 9
C                   EVAL      snSPP01Length = 3
C                   EVAL      saSPP01Unit = '*ROWCOL'
C                   EVAL      saSPP01Oper = '*NE'
C                   EVAL      svSPP01CmpStr = '*PRV'
C                   CALLP     AR_ADDITMR(wiRtnCode:
C                                        'CS_SPP01'     :
C                                        %size(CS_SPP01):
C                                        CS_SPP01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ****************************************************************
 * Add option structures for positional exit program parameters
 ****************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                   CS_EPP01
C                   EVAL      siEPP01PagNbr = 1
C                   EVAL      snEPP01Y = 5
C                   EVAL      snEPP01X = 9
C                   EVAL      snEPP01Length = 3
C                   EVAL      saEPP01Unit = '*ROWCOL'
C                   CALLP     AR_ADDITMR(wiRtnCode:
C                                        'CS_EPP01'     :
C                                        %size(CS_EPP01):
C                                        CS_EPP01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

C                   IF        not wlOptAPIError
C                   CLEAR                   CS_EPP01
C                   EVAL      siEPP01PagNbr = 1
C                   EVAL      snEPP01Y = 6
C                   EVAL      snEPP01X = 9
C                   EVAL      snEPP01Length = 5
C                   EVAL      saEPP01Unit = '*ROWCOL'
C                   CALLP     AR_ADDITMR(wiRtnCode:
C                                        'CS_EPP01'     :
C                                        %size(CS_EPP01):
C                                        CS_EPP01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
```

```
C                   ENDIF

C                   IF        not wlOptAPIError
C                   CLEAR                   CS_EPP01
C                   EVAL      siEPP01PagNbr = 1
C                   EVAL      snEPP01Y = 6
C                   EVAL      snEPP01X = 9
C                   EVAL      snEPP01Length = 30
C                   EVAL      saEPP01Unit = '*ROWCOL'
C                   CALLP     AR_ADDITMR(wiRtnCode:
C                                        'CS_EPP01'     :
C                                        %size(CS_EPP01):
C                                        CS_EPP01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ******************************************************************
 * Add option structure for exit program to supply stream file name
 * and set the passwords
 ******************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                   CS_EXT01
C                   EVAL      saExt01Pgm = 'EXITTYPE2'
C                   EVAL      saExt01Lib = '*LIBL'
C                   EVAL      saExt01Type = '*TYPE2'
C                   EVAL      saExt01When = '*STMFSTR'
C                   CALLP     AR_ADDITMR(wiRtnCode:
C                                        'CS_EXT01'     :
C                                        %size(CS_EXT01):
C                                        CS_EXT01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ******************************************************************
 * Add option structure for exit program to email the file
 ******************************************************************
C                   IF        not wlOptAPIError
C                   CLEAR                   CS_EXT01
C                   EVAL      saExt01Pgm = 'EXITTYPE1'
C                   EVAL      saExt01Lib = '*LIBL'
C                   EVAL      saExt01Type = '*TYPE1'
C                   EVAL      saExt01When = '*STMFEND'
C                   CALLP     AR_ADDITMR(wiRtnCode:
C                                        'CS_EXT01'     :
C                                        %size(CS_EXT01):
C                                        CS_EXT01       )
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ******************************************************************
 * Retrieve the length of the option structure list
 ******************************************************************
C                   IF        not wlOptAPIError
C                   CALLP     AR_RTVLENR(wiRtnCode  :
C                                        wuOptionDta)
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ******************************************************************
```

```
 * Allocate sufficient storage to hold the list
 ****************************************************************
C                   IF        not wlOptAPIError
C                   ALLOC     wuOptionDta   wpOptionDta
C                   ENDIF

 ****************************************************************
 * Retrieve the option structure list data
 ****************************************************************
C                   IF        not wlOptAPIError
C                   CALLP     AR_RTVDTAR(wiRtnCode  :
C                                        wuOptionDta:
C                                        waOptionDta:
C                                        wuOptionDta)
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF
C                   ENDIF

 ****************************************************************
 * Call the CoolSpools Spool Conversion API
 ****************************************************************
C                   IF        not wlOptAPIError
C                   CALLP     CS_CVTAPI('QSYSPRT ' :
C                                       '*'        :
C                                       -1         :
C                                       8          :
C                                       '*EXITPGM' :
C                                       '*PDF'     :
C                                       '*REPLACE' :
C                                       CS_ERR01   :
C                                       wuOptionDta:
C                                       waOptionDta:
C                                       CS_RTN01   )
C                   ENDIF

 ****************************************************************
 * Delete the option structure list
 ****************************************************************

C                   IF        not wlOptAPIError
C                   CALLP     AR_DLTLSTR(wiRtnCode)
C                   IF        wiRtnCode <> OPT_OK
C                   EVAL      wlOptAPIError = *ON
C                   ENDIF

C                   ENDIF

C                   SELECT

 ****************************************************************
 * If an error occurred calling the conversion API, get the error
 * details from the error structure. The available length returned
 * includes the length of the error message id and the following one-
 * byte reserved area (for consistency with IBM APIs).
 ****************************************************************
C                   WHEN      suErr01Avail > 0
C                   EVAL      wlError = *ON
C                   EVAL      waMsgId = saErr01MsgId
C                   EVAL      wuMsgDtaLen = suErr01Avail
C                   IF        suErr01Avail > %size(saErr01MsgId) + 1
C                   EVAL      wuMsgDtaLen = suErr01Avail
C                                         - %size(saErr01MsgId) - 1
C                   IF        wuMsgDtaLen > %size(waMsgDta)
C                   EVAL      wuMsgDtaLen = %size(waMsgDta)
C                   ENDIF
C                   EVAL      waMsgDta = %subst(saErr01MsgDta:
```

```
C                                          1             :
C                                          wuMsgDtaLen   )
C                   ELSE
C                   EVAL      wuMsgDtaLen = 0
C                   ENDIF

 ****************************************************************
 * If an error occurred calling an option structure list API, get the
 * error details
 ****************************************************************
C                   WHEN      wlOptAPIError
C                   EVAL      wlError = *ON
C                   CALLP     AR_RTVERRR(waMsgId        :
C                                        %size(waMsgDta):
C                                        waMsgDta       :
C                                        wuMsgDtaLen    )

C                   ENDSL

 ****************************************************************
 * Close down the option structure list environment. Do this even if
 * an error occurred previously.
 ****************************************************************
C                   IF        xlOptInitialized
C                   CALLP     AR_ENDOPTR(wiRtnCode)
C                   ENDIF

 ****************************************************************
 * Insert error handling appropriate to your environment here.
 * Here we just send an escape message which will abort processing.
 ****************************************************************
C                   IF        wlError

 * Set the message file name

C                   SELECT
C                   WHEN      %subst(waMsgId:1:3) = 'CVT'
C                   EVAL      waMsgFile = 'CP_MSGF   *LIBL'
C                   WHEN      %subst(waMsgId:1:3) = 'ARI'
C                   EVAL      waMsgFile = 'AR_MSGF   *LIBL'
C                   OTHER
C                   EVAL      waMsgFile = 'QCPFMSG   *LIBL'
C                   ENDSL

C                   CALLP     SndPgmMsg(waMsgId    :
C                                       waMsgFile  :
C                                       waMsgDta   :
C                                       wuMsgDtaLen:
C                                       '*ESCAPE'  :
C                                       '*PGMBDY'  :
C                                       1          :
C                                       waMsgKey   :
C                                       APIError   )
C                   ELSE

 ****************************************************************
 * Send a completion message
 ****************************************************************
C                   EVAL      waMsgId = 'CPF9898'
C                   EVAL      waMsgFile = 'QCPFMSG   *LIBL'
C                   EVAL      waMsgDta = %trim(%editc(suRtn01StmCnt:'Z'))
C                                      + ' stream files created'
C                   EVAL      wuMsgDtaLen = %len(%trimr(waMsgDta))
C                   CALLP     SndPgmMsg(waMsgId    :
C                                       waMsgFile  :
C                                       waMsgDta   :
C                                       wuMsgDtaLen:
```

```
C                                       '*COMP'    :
C                                       '*PGMBDY'  :
C                                       1          :
C                                       waMsgKey   :
C                                       APIError   )

C                   ENDIF

C                   EVAL      *INLR = *ON
C                   RETURN
```

CoolSpools Merge API

The CoolSpools Merge API invokes the CoolSpools functionality which merges PDF files. As such it provides an alternative to calling the CoolSpools command interface MRGPDF.

Where you wish to integrate PDF merging into your applications, the CoolSpools Spool Conversion API may provide a more convenient interface than running a command, especially if you need to interface into CoolSpools from code written in a language such as RPG, COBOL, C or Java, or if you need to specify complex parameters.

RPG Copybooks

A number of source members are provided in file CS_SRCFILE for use with ILE RPG. These can simplify the calling of the CoolSpools Merge API by providing data definitions.

The members are:

CS_MRGAPIP

This source member contains the definition of data structures and prototypes required to call the Merge API.

CoolSpools Merge API (CS_MRGAPIR)

The CoolSpools Spool Conversion API (*PGM object ***CS_MRGAPIR***) allows access to CoolSpools functionality to converts an iSeries spooled file to one of several different file formats.

Required parameter group			
1	Name of the first file to be merged	Input	CHAR(*)
2	Length of first file name	Input	BINARY(10)
3	Password of first file	Input	CHAR(32)
4	Name of second file to be merged	Input	CHAR(*)
5	Length of second file name	Input	BINARY(4)
6	Password of second file	Input	CHAR(32)
7	Name of file to be created	Input	CHAR(*)
8	Length of name of file to be created	Input	BINARY(4)
9	Replace if file exists	Input	CHAR(10)

Omissible parameter group 1			
10	Error structure	I-O	CHAR(*)
Omissible parameter group 2			
11	Format of password information	Input	CHAR(8)
12	Password information	Input	CHAR(*)
13	Length of password information	Input	BINARY(4)
Omissible parameter group 3			
14.	Public data authority of merged file	Input	CHAR(10)
Omissible parameter group 4			
15.	Page rotation of first file	Input	BINARY(4)
16.	Page rotation of second file	Input	BINARY(4)

Required Parameter Group

Option structure priority

INPUT; CHAR(1)

A code identifying the priority of the option structure being added to the list.

The following values are supported for this parameter:

1 Primary. So long as the page scope is appropriate, the option structure will always be selected and will influence CoolSpools processing.

2 Secondary. The option structure will only be selected and influence CoolSpools processing if the page scope is appropriate and no other structure of this format with primary priority has already been selected.

If this parameter is not passed, secondary priority is assumed.

Name of first file to be merged

INPUT; CHAR(*)

The name of the first PDF file to be included in the merged PDF file. This should be the full absolute or relative path name.

The second file named below will be appended to this file to create the merged file specified below.

Length of first file name

INPUT; BINARY(4)

The length of the file name defined on the previous parameter.

Password of first file

INPUT; CHAR(32)

The password to use when opening the first file.

If no password is required to open the file, blanks should be passed on this parameter.

If the file has been secured with a password, you will need to enter a password on this parameter otherwise it cannot be processed. If the file has been secured in such a way that its contents cannot be copied or modified without entering the owner password, you will be required to supply the owner password on this parameter to process the file.

Name of second file to be merged

INPUT; CHAR(*)

The name of the second PDF file to be included in the merged PDF file. This should be the full absolute or relative path name.

This file is appended to the first file named above to create the merged file specified below.

Length of second file name

INPUT; BINARY(4)

The length of the file name defined on the previous parameter.

Password of second file

INPUT; CHAR(32)

The password to use when opening the second file.

If no password is required to open the file, blanks should be passed on this parameter.

If the file has been secured with a password, you will need to enter a password on this parameter otherwise it cannot be processed. If the file has been secured in such a way that its contents cannot be copied or modified without entering the owner password, you will be required to supply the owner password on this parameter to process the file.

Name of merged PDF file

INPUT; CHAR(*)

The name of the merged PDF file to be created by combining the two files named above. This should be the full absolute or relative path name.

The following special value can be used:

**FROMPDF* The merged PDF file will have the same file name as the file specified as the first PDF file above. The second file will be appended to the end of the first file and then the first file will be replaced by the combined file thus created.

Length of second file name

INPUT; BINARY(4)

The length of the file name defined on the previous parameter.

Replace file if it exists

INPUT; CHAR(10)

Whether the merged PDF file should be replaced if it already exists.

**YES* Replace the file is it exists already.

**NO* Do not replace the file is it exists already. An error will occur if the file already exists and it will not be replaced.

Optional Parameter Group 1

Error code

I/O; CHAR(*)

The structure in which to return error information. The format of the structure is defined under “Error structure” below.

If this parameter is omitted, diagnostic and escape messages are issued to the application.

Error structure

The error structure conforms to the format of the standard IBM API structure.

This structure is called CS_ERR01 in CoolSpools copybooks.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Structure size	The size of the error structure	BINARY(4)
4	0004	Length of error data	The total length of the error data available in this structure.	BINARY(4)

8	0008	Error message	The message identifier of the error that occurred.	CHAR(7)
15	000F	Reserved	Reserved	CHAR(1)
16	0010	Error data	The substitution data associated with the error message	CHAR(*)

Optional Parameter Group 2

Format of password information

INPUT; CHAR(8)

The format of the password information specified on the following parameter. At present, the only valid value is CS_MRG01.

If this parameter is passed, you must specify the value CS_MRG01 and must also specify a valid CS_MRG01 structure on the following parameter.

Password information

INPUT; CHAR(*)

A structure defining the password(s) and/or security to be applied to the merged PDF file. This structure must conform to format CS_MRG01 as defined below.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Option	Password option. Valid values are: ***FROMPDF** = Passwords and security are replicated from the file specified on the first PDF file name parameter. ***NO** = No passwords or security are applied to the merged PDF file created. ***YES** = At least one password will be applied to the merged PDF file created. ***RESTRICT** = No passwords will be applied to the file but the operations that can be performed on it will be restricted.	CHAR(10)
10	000A	User password	The user password to apply.	CHAR(32)

			Specify blanks if no user password required.	
42	002A	Owner password	The owner password to apply. Specify blanks if no owner password required.	CHAR(32)
74	004A	Allow printing	Whether the file may be printed. ***YES** = Allow printing. ***NO** = Disallow printing.	CHAR(10)
84	0054	Allow modifications	Whether the file may be modified. ***YES** = Allow modifications. ***NO** = Disallow modifications.	CHAR(10)
94	005E	Allow copying of text	Whether text in the file may be copied. ***YES** = Allow copying of text. ***NO** = Disallow copying of text.	CHAR(10)
104	0068	Allow annotations	Whether the file may be annotated. ***YES** = Allow annotations ***NO** = Disallow Annotations	CHAR(10)

Length of password information

INPUT; BINARY(4)

The size of the structure passed on the previous parameter.

Optional Parameter Group 3

Public data authorities for merged file

INPUT; CHAR(10)

The public data authority to give to the merged PDF file.

If no value is specified for this parameter, the authorities are copied from the first PDF file specified above.

Options are:

*FROMPDF Copy the authorities from the first PDF file named above.

*R Read only

*W	Write only
*X	Execute only
*RW	Read and write
*RX	Read and execute
*WX	Write and execute
*RWX	Read, write and execute (all)
*NONE	No authority

Optional Parameter Group 4

Page rotation for first PDF file

INPUT; BINARY(4)

The page rotation to apply to the first PDF file, in degrees.

If the page size or orientation of the two files being merged differs, it can sometimes be convenient to rotate the pages of one of the files in order to make the combined file easier to read. Specify an angle through which the pages in the file should be rotated.

Valid values are: 0, 90, 180 and 270.

If this parameter is not passed, 0 is assumed.

Page rotation for second PDF file

INPUT; BINARY(4)

The page rotation to apply to the second PDF file, in degrees.

If the page size or orientation of the two files being merged differs, it can sometimes be convenient to rotate the pages of one of the files in order to make the combined file easier to read. Specify an angle through which the pages in the file should be rotated.

Valid values are: 0, 90, 180 and 270.

If this parameter is not passed, 0 is assumed.

Example

The following example program demonstrates how to call the CoolSpools merge API

The source of this example program is supplied in CoolSpools source file CS_SRCFILE.

If should be created as follows.

```
CRTBNDRPG PGM(MRGEXAMPLE) SRCFILE(CS_SRCFILE)
```

```
*************************************************************
*                                                           *
* PROGRAM NAME: MRGEXAMPLE                                  *
```

```
 *                                                                 *
 * DESCRIPTION : Demonstrates how to call the CoolSpools Merge     *
 *               API from ILE RPG.                                 *
 *                                                                 *
 *******************************************************************

H EXTBININT

 /COPY CS_SRCFILE,CS_MRGAPID

 /COPY CS_SRCFILE,CS_MRGAPIP

 * Prototype for API to send a program message
D SndPgmMsg       PR                  EXTPGM('QMHSNDPM')
D  iaMsgId                       7A   CONST
D  iaMsgFile                    20A   CONST
D  iaMsgDta                  32767A   CONST OPTIONS(*VARSIZE)
D  iiMsgDtaLen                  10I 0 CONST
D  iaMsgType                    10A   CONST
D  iaStackEntry                 10A   CONST
D  iiStackCount                 10I 0 CONST
D  oaMsgKey                      4A
D  baAPIError                 1024A         OPTIONS(*VARSIZE)

 * IBM API error structure
D APIError        DS
D  siAPIErrSize                 10I 0 INZ(%size(APIError))
D  siAPIErrLen                  10I 0 INZ(0)
D  saAPIErrMsg                   7A
D                                1A
D  saAPIErrDta                 256A

 * Error message data
D waMsgDta        S             1024A

 * Length of associated error message data
D wuMsgDtaLen     S               10U 0

 * Qualified message file name
D waMsgFile       S               20A

 * Message key returned from SNDPGMMSG API
D waMsgKey        S                4A

D wvPDF1          S             1024A   VARYING
D wvPDF2          S             1024A   VARYING
D wvToPDF         S             1024A   VARYING

 *******************************************************************
 * Set up the error structure
 *******************************************************************
C                   CLEAR                   CS_ERR01
C                   EVAL      suErr01Size = %size(CS_ERR01)

C                   CLEAR                   CS_MRG01
C                   EVAL      saMrg01Option = '*NO'

C                   EVAL      wvPDF1 = 'TEST1.PDF'
C                   EVAL      wvPDF2 = 'TEST2.PDF'
C                   EVAL      wvToPDF = 'MERGED.PDF'
C                   CALLP     CS_MrgAPI(wvPDF1        :
C                                       %len(wvPDF1)   :
C                                       *BLANKS        :
C                                       wvPDF2         :
C                                       %len(wvPDF2)   :
C                                       *BLANKS        :
C                                       wvToPDF        :
```

```
C                                       %len(wvToPDF)   :
C                                       '*YES'          :
C                                       CS_ERR01        :
C                                       'CS_MRG01'      :
C                                       CS_MRG01        :
C                                       %size(CS_MRG01):
C                                       '*R'            :
C                                       90              :
C                                       90              )

 ****************************************************************
 * Insert error handling appropriate to your environment here.
 * Here we just send an escape message which will abort processing.
 ****************************************************************
C                   IF        suErr01Avail > 0

 * Set the message file name

C                   SELECT
C                   WHEN      %subst(saErr01MsgId:1:3) = 'CVT'
C                   EVAL      waMsgFile = 'CP_MSGF   *LIBL'
C                   WHEN      %subst(saErr01MsgId:1:3) = 'ARI'
C                   EVAL      waMsgFile = 'AR_MSGF   *LIBL'
C                   OTHER
C                   EVAL      waMsgFile = 'QCPFMSG   *LIBL'
C                   ENDSL

C                   EVAL      wuMsgDtaLen = suErr01Avail
C                   IF        suErr01Avail > %size(saErr01MsgId) + 1
C                   EVAL      wuMsgDtaLen = suErr01Avail
C                                         - %size(saErr01MsgId) - 1
C                   IF        wuMsgDtaLen > %size(waMsgDta)
C                   EVAL      wuMsgDtaLen = %size(waMsgDta)
C                   ENDIF
C                   EVAL      waMsgDta = %subst(saErr01MsgDta:
C                                               1            :
C                                               wuMsgDtaLen  )
C                   ELSE
C                   EVAL      wuMsgDtaLen = 0
C                   ENDIF

C                   CALLP     SndPgmMsg(saErr01MsgId:
C                                       waMsgFile   :
C                                       waMsgDta    :
C                                       wuMsgDtaLen :
C                                       '*ESCAPE'   :
C                                       '*PGMBDY'   :
C                                       1           :
C                                       waMsgKey    :
C                                       APIError    )
C                   ELSE

 ****************************************************************
 * Send a completion message
 ****************************************************************
C                   EVAL      saErr01MsgId = 'CPF9898'
C                   EVAL      waMsgFile = 'QCPFMSG   *LIBL'
C                   EVAL      waMsgDta = 'PDF files merged'
C                   EVAL      wuMsgDtaLen = %len(%trimr(waMsgDta))
C                   CALLP     SndPgmMsg(saErr01MsgId    :
C                                       waMsgFile  :
C                                       waMsgDta   :
C                                       wuMsgDtaLen:
C                                       '*COMP'    :
C                                       '*PGMBDY'  :
C                                       1          :
C                                       waMsgKey   :
```

```
C                                       APIError   )

C                   ENDIF

C                   EVAL      *INLR = *ON
C                   RETURN
```

CoolSpools Exit Programs

One of CoolSpools' most powerful features is its ability to call exit programs that can influence its processing during the course of a spooled file conversion. Exit programs allow you to override conversion parameters at run time, setting attributes such as the name of the stream file to be created, the email addresses to which the file should be emailed and even whether particular pages should be included or excluded.

Defining an Exit Program to call

There are three ways in which you can specify the name of an exit program to be called:

- On the EXITPGM parameter of the CVTSPLSTMF (Convert Spooled File to Stream File) command. Please note however that for reasons of backwards compatibility with previous releases, this parameter only allows the definition of a single exit program at one of only four exit points (*STMFSTR, *STMFEND, *PAGEEND or *SPLFEND).
- On the EXITPGM parameter of one of the format-specific commands (CVTSPLPDF, CVTSPLXLS etc.). This parameter allows the definition of up to 100 exit programs at any of the exit points listed below.
- By adding CS_EXT01 structures to the option structure list before calling the CoolSpools Spool Conversion API. Up to 100 such structures may be active at any one time.

If multiple exit programs are defined at the same exit point, CoolSpools will call them in the order in which they are defined.

Exit Points

The following exit points are currently defined.

- ***SPLFSTR** Start of Spooled File. This exit point is processed before any data is output, i.e. before the first stream file is opened. This might be a convenient place to do startup and initialization tasks prior to the conversion starting.
- ***STMFSTR** Start of Stream File. This exit point is processed after *SPLFSTR for the first stream file and before each stream file is opened. This is typically where you will set attributes for the stream file about to be created, e.g. the stream file name.
- ***SHEETSTR** Start of a new worksheet. This exit point is processed just before a new worksheet is created when converting to Excel format.
- ***PAGECTL** Page Control. This exit point is intended solely to provide an opportunity to generate a CS_FBK01 option structure to indicate whether a particular page should be included or excluded. It is processed for each page in the stream file before the first *PAGESTR exit point is processed to give the application an opportunity to select the pages to be output before any of the pages is output.
- ***PAGESTR** Start of Page. This exit point is processed before any data is output for each page. In relation to the first page of the stream file, it will be processed after *STMFSTR and after *PAGECTL has been processed for each page in the stream file. Please note however that pages may not be processed in page number order. In particular, when creating a PDF file, the first page in the file is processed last, for reasons related to the structure of a web-optimized PDF file.
- ***PAGEEND** End of Page. This exit point is processed after all data has been output for each page. Please note however that pages may not be processed in page number order. In particular, when creating a PDF file, the first page in the file is processed last, for reasons related to the structure of a web-optimized PDF file.
- ***STMFEND** End of Stream File. This exit point is processed after *PAGEEND for the last page in each stream file and before *SPLFSTR for the last stream file.
- ***SPLFEND** End of Spooled File. This exit point is processed after *STMFEND for the last stream file. This may be a convenient place in which to do housekeeping at the end of the conversion run.
- ***SHEETEND** End of a worksheet. This exit point is processed just after a worksheet is finished when converting to Excel format.

Writing an Exit Program

Exit programs can be written in any language which can be compiled into a system i *PGM object.

However, if you want to use the option structure list APIs to interact with CoolSpools options at run time, you will need to use an ILE language to call them. In order to modify the CoolSpools options, the exit program must run in the same activation group as CoolSpools and this is not possible for an OPM program.

CoolSpools Exit Programs are passed a standard parameter list in one of four formats.

- ***TYPE1** Type 1 parameter list. This is the original parameter list format used by exit programs when the exit program function was introduced in Version 2. It is provided for reasons of backwards compatibility. You are recommended to use *TYPE3 or *TYPE4 for future applications.
- ***TYPE2** Type 2 parameter list. This is the parameter list format introduced by Version 3 when the ability to pass more than one user-definable exit program parameter to exit programs was introduced. It is provided for reasons of backwards compatibility. You are recommended to use *TYPE3 for future applications.
- ***TYPE3** Type 3 parameter list. This is a new parameter list format introduced by Version 6. You are recommended to use *TYPE3 for future applications. It is very similar to *TYPE2 but includes some additional features which can be useful when developing applications that use exit programs.
- ***TYPE4** Type 4 parameter list. *TYPE4 is recommended for applications that need to process a large number of user-defined parameters since there is no practical limit to the number of user-defined parameters the *TYPE4 method can handle (the limit for *TYPE3 is 242 user-defined parameters because of the IBM I (OS/400) limit of 256 parameters passed to a program by the CALL command).

Please note that *TYPE3 and *TYPE4 programs have an additional feature which does not apply to *TYPE1 or *TYPE2 exit programs. As well as being called at their normal exit point, they are called once again at the end of processing with the exit point parameter set to *END. This is intended to provide a means of notifying the exit program that processing is complete and give it an opportunity to perform housekeeping tasks such as closing files.

The format of the different parameter lists is specified below. Copybooks CS_EXTTP1P, CS_EXTTP2P, CS_EXTTP3P and CS_EXTTP4P in source file CS_SRCFILE can be used to define these parameter lists in an exit program written in ILE RPG. In addition, source member CS_EXTTP4D defines the structures used by *TYPE4.

Type 1 parameter list

The Type 1 parameter list format was the only format available with exit programs in V2 of CoolSpools. You are still able to use this format, but it is supported primarily for reasons of backwards-compatibility. It has the limitation that it can receive only a single exit program parameter extracted from the spooled files, whereas V3 and later versions of CoolSpools can pass multiple exit program parameters to called exit programs.

A Type 1 parameter list consists of the following:

1	Spooled file name	Input	CHAR(10)
2	Name of the job that created the spooled file	Input	CHAR(10)
3	User id that created the spooled file	Input	CHAR(10)
4	Number of the job that created the spooled file	Input	CHAR(6)
5	Spooled file number	Input	BINARY(4)
6	Spooled file user data	Input	CHAR(10)
7	Name of the stream file just created	Input	CHAR(128)
8	Directory name in which stream file was created	Input	CHAR(256)
9	User-definable exit program parameter	Input	CHAR(1024)

Note that parameter 9 is a 1024-byte fixed length character variable which will be padded with trailing blanks if the parameter extracted from the spooled file is less than 1024 bytes long.

If more than one parameter is extracted from the spooled file, only the first parameter will be passed to an exit program defined as having *TYPE1 parameters. It is therefore recommended that you use *TYPE2 or *TYPE3 parameters if you wish to extract more than one parameter string from the spooled file. If you need to pass a large number of user-defined parameters (in excess of 242), you will need to use the new *TYPE4 parameter list.

Type 2 parameter list

The Type 2 parameter list format was introduced with Version 3 in order to allow multiple exit program parameters to be passed to called exit programs.

A Type 2 parameter list consists of the following items:

1	Spooled file name	Input	CHAR(10)
2	Name of the job that created the spooled file	Input	CHAR(10)

3	User id that created the spooled file	Input	CHAR(10)
4	Number of the job that created the spooled file	Input	CHAR(6)
5	Spooled file number	Input	BINARY(4)
6	Spooled file user data	Input	CHAR(10)
7	Name of the stream file just created	Input	CHAR(128)
8	Directory name in which stream file was created	Input	CHAR(256)
9	First page processed	Input	BINARY(4)
10	Last page processed	Input	BINARY(4)
11	Number of following user-definable parameters	Input	BINARY(4)
12	User definable parameter 1	Input	CHAR(*)
13	User definable parameter 2	Input	CHAR(*)
…			
	User definable parameter 200	Input	CHAR(*)

The number of user definable parameters that is passed on the call is indicated by the value of parameter 11.

This is followed by between 0 and 200 variable-length strings, the length of which ranges from 0 to 1024 bytes. These are equivalent to ILE RPG character variables with the VARYING attribute specified, i.e. they consist of a two-byte binary length followed by up to 1024 bytes of text.

Type 3 parameter list

The Type 3 parameter list format was introduced with Version 6 in order to provide additional information to exit programs which is often necessary for them to be able to do their job efficiently. Specifically, in addition to the items passed for a *TYPE2 parameter list, it includes the exit point at which the call is taking place and the format to which the stream file is being converted.

Please note that *TYPE3 programs have an additional feature which does not apply to *TYPE1 or *TYPE2 exit programs. As well as being called at their normal exit point, they are called once again at the end of processing with the exit point parameter set to *END. This is intended to provide a means of notifying the exit program that processing is complete and give it an opportunity to perform housekeeping tasks such as closing files.

A Type 3 parameter list consists of the following items:

1	Spooled file name	Input	CHAR(10)

2	Name of the job that created the spooled file	Input	CHAR(10)
3	User id that created the spooled file	Input	CHAR(10)
4	Number of the job that created the spooled file	Input	CHAR(6)
5	Spooled file number	Input	BINARY(4)
6	Spooled file user data	Input	CHAR(10)
7	Name of the stream file just created	Input	CHAR(128)
8	Directory name in which stream file was created	Input	CHAR(256)
9	First page processed	Input	BINARY(4)
10	Last page processed	Input	BINARY(4)
11	Exit point being processed	Input	CHAR(10)
12	Format to which the data is being converted	Input	CHAR(10)
13	Number of following user-definable parameters	Input	BINARY(4)
14	User definable parameter 1	Input	CHAR(*)
15	User definable parameter 2	Input	CHAR(*)
…			
	User definable parameter 200	Input	CHAR(*)

The number of user definable parameters that is passed on the call is indicated by the value of parameter 11.

This is followed by between 0 and 200 variable-length strings, the length of which ranges from 0 to 1024 bytes. These are equivalent to ILE RPG character variables with the VARYING attribute specified, i.e. they consist of a two-byte binary length followed by up to 1024 bytes of text.

Type 4 parameter list

The Type 4 parameter list will be fully rolled out in the next release, but was added to Version 6 in relation to PDF output by PTFs 5CV0200 (CoolSpools) and 5CP0095 (CoolSpools PLUS). *TYPE4 is recommended for applications that need to process a large number of user-defined parameters since there is no practical limit to the number of user-defined parameters the *TYPE4 method can handle (the limit for *TYPE3 is 242 user-defined parameters because of the IBM I (OS/400) limit of 256 parameters passed to a program by the CALL command).

A Type 4 parameter list consists of a single program parameter structure in one of two formats (additional formats may be introduced in future versions):

CS_EPC01	Exit Program Call parameters for all types of output other than *SPLF (CVTSPLSPLF command).
CS_EPC02	Exit Program Call parameters for output type *SPLF (CVTSPLSPLF command).

Structure CS_EPC01 – Exit Program Call parameters (stream file output)

Name	Description
CS_EPC01	Contains the information passed from CoolSpools to an exit program when the output type is anything other than *SPLF (spooled file – CVTSPLSPLF command).

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Format	The format of this structure (CS_EPC01)	CHAR(8)
8	0008	Length	Total length of this structure in bytes	BINARY(4)
12	000C	Exit point	The name of the exit point at which this call to the exit program is occurring.	CHAR(10)
22	0016	To-format	The format to which the spooled file is being converted.	CHAR(10)
32	0020	From page	The first page in the spooled file that is being converted to which this call relates. Where a spooled file is being split, and the exit point is one of the split file exit points (*STMFSTR, *STMFEND), or if the exit point is a page-level exit point, this indicates the first page number in the range of pages being processed.	BINARY(4)
36	0024	To page	The last page in the page range being processed. See above.	BINARY(4)
40	0028	CCSID	The CCSID of the stream file and	BINARY(4)

			directory path information following	
44	003C	Length of stream file name	The length of the stream file name following.	BINARY(2)
46	003E	Stream file name	The name of the stream file that was created or will be to be created (timing depends on exit point).	CHAR(128)
174	00AE	Length of directory path	The length of the directory path following.	BINARY(2)
176	00B0	Directory path	The directory path in which the stream file was created or will be created (timing depends on exit point).	CHAR(256)
432	01B0	Offset to spooled file information	The byte offset from the start of this structure to the start of the spooled file information.	BINARY(4)
436	01B4	Length of spooled file information	The length of the spooled file information section, in bytes.	BINARY(4)
440	01B8	Format of spooled file information	The format of the spooled file information section (SPLA0200)	CHAR(8)
448	01C0	Number of user-defined parameters	The number of user-defined parameters that follow the spooled file information section. These correspond to the exit program parameters extracted using the EXITPGMPOS and/or EXITPGMKEY parameters of the CVTSPLPDF command etc., or defined using the CS_EPP01 or CS_EPK01 option structures.	BINARY(4)
452	01C4	CCSID of user-defined parameter information	The CCSID in which the user-defined parameter structures that follow are encoded.	BINARY(4)
456	01C8	Offset to user-defined	The offset from the start of this structure to the first user-defined	BINARY(4)

		parameters	parameter structure.	
460	01CC	Length of user-defined parameters.	The total length in bytes of the user-defined parameter section.	BINARY(4)
464	01D0	Format of user-defined parameter structures.	The format of each user-defined parameter structure. CS_UDP01 or CS_UDP02 depending on the value of environment variable CS_EXIT_PGM_TYPE4_UDP_FORMAT	CHAR(8)
Variable. Refer to “Offset to spooled file information” field above.		Spooled file information	The spooled file information section. This provides a full set of attributes of the spooled file being converted and is currently supplied in the SPLA0200 format returned by the QUSRSPLA API. See http://publib.boulder.ibm.com/system i/v5r1/ic2924/index.htm?info/apis/QUSRSPLA.htm	Variable. Refer to “Length of spooled file information” field above.
Variable. Refer to “Offset to user-defined parameters” field above.		User-defined parameter section.	User-defined parameters section. This consists of an array of CS_UDP01 or CS_UDP02 structures depending on the value of exit program CS_EXIT_PGM_TYPE4_UDP_FORMAT Each structure corresponds to an exit program parameter extracted using the EXITPGMPOS and/or EXITPGMKEY parameters of the CVTSPLPDF command etc., or defined using the CS_EPP01 or CS_EPK01 option structures.	Variable. Refer to “Length of user-defined parameters” field above.

Structure CS_EPC02 – Exit Program Call parameters (spooled file output)

Name	Description
CS_EPC02	Contains the information passed from CoolSpools to an exit program when the output type is *SPLF (spooled file –

	CVTSPLSPLF command).

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	Format	The format of this structure (CS_EPC02)	CHAR(8)
8	0008	Length	Total length of this structure in bytes	BINARY(4)
12	000C	Exit point	The name of the exit point at which this call to the exit program is occurring.	CHAR(10)
22	0016	To-format	The format to which the spooled file is being converted.	CHAR(10)
32	0020	From page	The first page in the spooled file that is being converted to which this call relates. Where a spooled file is being split, and the exit point is one of the split file exit points (*STMFSTR, *STMFEND), or if the exit point is a page-level exit point, this indicates the first page number in the range of pages being processed.	BINARY(4)
36	0024	To page	The last page in the page range being processed. See above.	BINARY(4)
40	0028	Spooled file name	The name of the spooled file that was created or will be to be created (timing depends on exit point).	CHAR(10)
50	0032	Offset to input spooled file information	The byte offset from the start of this structure to the start of the spooled file information for the input spooled file (the spooled file being converted)	BINARY(4)
54	0036	Length of input spooled file information	The length of the spooled file information section, in bytes, for the input spooled file (the spooled file being converted)	BINARY(4)

58	003A	Format of input spooled file information	The format of the spooled file information section (SPLA0200) for the input spooled file (the spooled file being converted)	CHAR(8)
66	0042	Offset to output spooled file information	The byte offset from the start of this structure to the start of the spooled file information for the output spooled file (the spooled file that was created). This is zero if the exit point occurs prior to the creation of the output spooled file.	BINARY(4)
70	0046	Length of output spooled file information	The length of the spooled file information section, in bytes, for the output spooled file (the spooled file that was created). This is zero if the exit point occurs prior to the creation of the output spooled file.	BINARY(4)
74	004A	Format of output spooled file information	The format of the spooled file information section (SPLA0200) for the output spooled file (the spooled file that was created). This is zero if the exit point occurs prior to the creation of the output spooled file.	CHAR(8)
82	0052	Number of user-defined parameters	The number of user-defined parameters that follow the spooled file information section. These correspond to the exit program parameters extract using the EXITPGMPOS and/or EXITPGMKEY parameters of the CVTSPLPDF command etc., or defined using the CS_EPP01 or CS_EPK01 option structures.	BINARY(4)
86	0056	CCSID of user-defined parameter information	The CCSID in which the user-defined parameter structures that follow are encoded.	BINARY(4)
90	005A	Offset to user-defined parameters	The offset from the start of this structure to the first user-defined parameter structure.	BINARY(4)

94	005E	Length of user-defined parameters.	The total length in bytes of the user-defined parameter section.	BINARY(4)
Variable. Refer to “Offset to input spooled file information” field above.		Input spooled file information	The spooled file information section. This provides a full set of attributes of the spooled file being converted and is currently supplied in the SPLA0200 format returned by the QUSRSPLA API. See http://publib.boulder.ibm.com/system i/v5r1/ic2924/index.htm?info/apis/QUSRSPLA.htm	Variable. Refer to “Length of input spooled file information” field above.
Variable. Refer to “Offset to output spooled file information” field above.		Output spooled file information	The spooled file information section. This provides a full set of attributes of the spooled file just created and is currently supplied in the SPLA0200 format returned by the QUSRSPLA API. See http://publib.boulder.ibm.com/system i/v5r1/ic2924/index.htm?info/apis/QUSRSPLA.htm This section is absent if the exit program is being called prior to the creation of the output spooled file.	Variable. Refer to “Length of output spooled file information” field above.
Variable. Refer to “Offset to user-defined parameters” field above.		User-defined parameter section.	User-defined parameters section. This consists of an array of CS_UDP01 or CS_UDP02 structures, depending on the value of environment variable CS_EXIT_PGM_TYPE4_UDP_FORMAT Each structure corresponds to an exit program parameter extracted using the EXITPGMPOS and/or EXITPGMKEY parameters of the CVTSPLPDF command etc., or defined using the CS_EPP01 or CS_EPK01 option structures.	Variable. Refer to “Length of user-defined parameters” field above.

Structure CS_UDP01 –User-defined Parameter

Name	Description
CS_UDP01	User-defined parameter value extracted from the spooled file by use of the EXITPGMPOS or EXITPGMKEY command parameters, or by use of the CS_EPP01 or CS_EPK01 option structures. The user-defined parameters section of the CS_EPC01 and CS_EPC02 structures consists of an array of 0 or more of these structures. This is the default structure used to pass user-defined parameters to *TYPE4 exit programs and is used unless environment variable CS_EXIT_PGM_TYPE4_UDP_FORMAT is set to CS_UDP02.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	User-defined parameter number	The number of this user-defined parameter. A unique sequential number that identifies this structure in the list of user-defined parameter structures.	BINARY(4)
4	0004	Length	Total length of this structure in bytes, including the variable-length user-defined parameter data.	BINARY(4)
8	0008	Source	The source of this user-defined parameter. Possible values are: P = Positional exit program user-defined parameter defined on an EXITPGMPOS command parameter or CS_EPP01 option structure. K = Key string exit program user-defined parameter defined on an EXITPGMKEY command parameter or CS_EPK01 option structure.	CHAR(1)
9	0009	Page number	The page number in the spooled file being converted from which the data was extracted.	BINARY(4)
13	000D	User-defined parameter	The number of the EXITPGMPOS or EXITPGMKEY parameter, or	BINARY(4)

		sequence number	the CS_EPP01 or CS_EPK01 structure, which generated this data. For example, if two user-defined parameters are defined on the EXITPGMPOS parameter, the data related to the first will have a 1 in this structure subfield and the other will have a 2 in this structure subfield.	
17	0011	Length of user-defined parameter data	The length in bytes of the data that follows	BINARY(4)
21	0015	Reserved	Reserved.	CHAR(3)
24	0016	User-defined parameter data.	The data extracted from the spooled file.	Variable.

Structure CS_UDP02 –User-defined Parameter

Name	Description
CS_UDP02	User-defined parameter value extracted from the spooled file by use of the EXITPGMPOS or EXITPGMKEY command parameters, or by use of the CS_EPP01 or CS_EPK01 option structures. The user-defined parameters section of the CS_EPC01 and CS_EPC02 structures consists of an array of 0 or more of these structures. This is the alternative structure used to pass user-defined parameters to *TYPE4 exit programs and is used if environment variable CS_EXIT_PGM_TYPE4_UDP_FORMAT is set to CS_UDP02. It contains largely the same information as CS_UDP01, with two major exceptions: • The parameter data itself starts at a variable start position defined by an offset from the start of the structure. This is intended to allow new fields to be inserted prior to the data, if necessary in the future. • The structure contains information relating to the location and dimensions on the page of the text item selected. This can be useful if the exit program needs to do things like positioning images on the page based on the parameters selected.

Offset (dec)	Offset (hex)	Name	Description	Format
0	0000	User-defined parameter number	The number of this user-defined parameter. A unique sequential number that identifies this structure in the list of user-defined parameter structures.	BINARY(4)
4	0004	Length	Total length of this structure in bytes, including the variable-length user-defined parameter data.	BINARY(4)
8	0008	Source	The source of this user-defined parameter. Possible values are: P = Positional exit program user-defined parameter defined on an EXITPGMPOS command parameter or CS_EPP01 option structure. K = Key string exit program user-defined parameter defined on an EXITPGMKEY command parameter or CS_EPK01 option structure.	CHAR(1)
9	0009	User-defined parameter sequence number	The number of the EXITPGMPOS or EXITPGMKEY parameter, or the CS_EPP01 or CS_EPK01 structure, which generated this data. For example, if two user-defined parameters are defined on the EXITPGMPOS parameter, the data related to the first will have a 1 in this structure subfield and the other will have a 2 in this structure subfield.	BINARY(4)
13	000D	Page number	The page number in the spooled file being converted from which the data was extracted.	BINARY(4)
17	0011	Line number	The line number in the spooled file being converted from which the data was extracted.	BINARY(4)
21	0015	Column	The column number in the spooled file being converted from	BINARY(4)

		number	which the data was extracted.	
25	0019	Y Coordinate	Vertical start position on page of text in points	DEC(7,2)
29	001D	X Coordinate	Horizontal start position on page of text in points	DEC(7,2)
33	0021	Text length	Length of text on page in points	DEC(7,2)
37	0025	Text height	Height of text on page in points	DEC(7,2)
41	0029	Offset of user-defined parameter data	Offset to the user-defined parameter data from the start of the structure	BINARY(4)
45	002D	Length of user-defined parameter data	The length in bytes of the data that follows	BINARY(4)
Variable		User-defined parameter data.	The data extracted from the spooled file.	Variable

User-defined Exit Program Parameters

There are two types of user-defined exit program parameters, *positional* and *key* parameters.

A ***positional*** parameter is extracted from the spooled file and passed to an exit program by specifying the position on the page at which the text to be extracted is located. This is most suitable for those text items which always appear at the same position on each page.

For example, if you know that the customer number always appears at column 3 of line 6 of your page, you can tell CoolSpools to extract the customer number from the spooled file and pass it to your exit program as a parameter by defining it as a positional exit program parameter.

A ***key*** parameter is extracted from the spooled file and passed to an exit program by specifying its position in relation to a *key string*, i.e. an item of text which appears on the page and in relation to which the parameter text always appears at a regular offset.

For example, if you know that the customer number can appear on different lines on your page, it cannot be extracted as a positional exit program parameter. However, if it always appears at a fixed position from the label “Customer number:”, you can use this label as a key string and tell CoolSpools to extract the customer number from

the spooled file and pass it to your exit program as a parameter by defining it as a key exit program parameter.

When running the CVTSPLSTMF (Convert Spooled File to Stream File) command or one of the format-specific commands (CVTSPLPDF, CVTSPLXLS etc.), you must indicate what type of parameters you wish to extract from the spooled file using the **EXITPGMPRM** parameter. This ensures that you are prompted for the right command parameters.

Positional exit program parameters are defined using the **EXITPGMPOS** parameter of the CVTSPLSTMF (Convert Spooled File to Stream File) or format-specific commands (CVTSPLPDF, CVTSPLXLS etc.).

Key exit program parameters are defined using the **EXITPGMKEY** parameter of the CVTSPLSTMF (Convert Spooled File to Stream File) command or format-specific commands (CVTSPLPDF, CVTSPLXLS etc.).

You can also define positional exit program parameters by adding CS_EPP01 option structures to your option structure list before calling the CoolSpools Spool Conversion API.

You can also define key exit program parameters by adding CS_EPK01 option structures to your option structure list before calling the CoolSpools Spool Conversion API.

The number of user definable parameters passed to your exit program is always equal to the number of user definable parameters specified on the **EXITPGMPOS** and **EXITPGMKEY** command parameters or specified by means of the CS_EPP01 and CS_EPK01 option structures. If no text is selected for one of these parameters (e.g. the key value specified does not appear), a zero-length parameter is passed.

Parameters are always passed in the order in which they are defined on the **EXITPGMPOS** and **EXITPGMKEY** parameters or the order in which CS_EPP01 and CS_EPK01 structures were generated. Parameters defined on **EXITPGMPOS** or CS_EPP01 precede those defined on **EXITPGMKEY** or CS_EPK01.

The “first page processed” and “last page processed” parameters indicate the first and last page processed since the last exit point of this type in question was reached. For an exit program called at the end-of-page exit point, these will both be equal to the page number in the original spooled file just processed. For a stream-file-creation exit point, these will be the page numbers in the original spooled file of the first and last pages that appear in the stream file just created. For an end-of-file exit program, these will be the first and last pages in the spooled file.

Uses of Exit Programs

Exit programs can be used for a variety of purposes. Some of these are explored below.

Exit programs called at the ***STMFSTR** (Start of Stream File) exit point are ideal for setting various attributes related specifically to the stream file about to be processed.

For example, an exit program might add a CS_STM01 option structure to the option structure list, scoped to the range of pages covered by the stream file, in order to set the stream file name to be something specific to the contents of the file about to be produced. The name could be based on the value of an exit program parameter

extracted from the spooled file containing, say, the customer number or order number.

Note that this could also be achieved through the use of the <:EXITPGMPOSn:> or <:EXITPGMKEYn:> CoolSpools parameters. See the User Guide for details.

Similarly, you might want to include a graphic specific to the file in question. You could do this by adding a CS_INC01 option structure to the option structure list specifying the name of a JPEG image appropriate for the customer in question (e.g. their logo), identified by means of a user-definable parameter containing the customer number.

Other attributes you might consider modifying at this exit point include the security settings and passwords of PDF files, (CS_PWD01 option structure), email attributes (CS_EML01) and email recipients (CS_EMT01).

The *STMFEND exit point, on the other hand, is a good place at which to do post-processing operations on the file just created. For example, you might want to move it to a different directory, rename it, email it (if not done by CoolSpools), send it to a server using FTP or interface into a document management system. All of these things and many more are possible by calling an exit program.

If you have an application and you are not sure if or how it can be done with CoolSpools, please contact support@ariadnesoftware.co.uk and we will be glad to assist you. Often we will be able to supply sample exit program source code free of charge.

Examples

Emailing the stream file just created using CoolSpools Email

In this first example, exit program EXITTYPE1 uses a “Type 1” (*TYPE1) parameter list.

It is intended to be called each time a new stream file is generated in order to e-mail the stream file just created to the customer to whom the document relates using ariadne’s CoolSpools Email product.

The exit program is defined to CVTSPLSTMF using:

EXITPGM(*LIBL/EXITTYPE1 *TYPE1

or, for one of the format specific commands (CVTSPLPDF etc.):

EXITPGM((*LIBL/EXITTYPE1 *TYPE1 *STMFEND))

It is assumed that the **EXITPGMPRM(*POS)** or **EXITPGMPRM(*KEY)** option has been used to extract information from the report which is then used to generate the email parameters.

```
*****************************************************************
*                                                               *
* PROGRAM NAME: EXITTYPE1                                       *
*                                                               *
* DESCRIPTION : Example of a CoolSpools exit program  in ILE RPG *
*                                                               *
*               Demonstrates how to call CoolSpools Email API to *
*               email a stream file.                            *
*                                                               *
*               N.B. Specify *TYPE1 for parameter type!!!       *
*                                                               *
*               Can be created with CRTBNDRPG.                  *
*                                                               *
*****************************************************************

 * Include the CoolSpools Email API structures and prototypes

 /COPY CM_SRCFILE,CM_SNDAPIP

 * Array of "to" structures

D aaTo            S                   LIKE(CM_ToStruct)
D                                     DIM(32)

 * Array of attachment structures

D aaAttach        S                   LIKE(CM_AttStruct)
D                                     DIM(32)

 * Returned email message id

D waEmailId       S             30A


D waOldName       S           1024A   VARYING
D waNewName       S           1024A   VARYING
D waNewFull       S           1024A   VARYING
D wuAttachments   S             10U 0
D wuRecipients    S             10U 0
```

```
D QUOTE           C                   ''''


 *****************************************************************
 * PROTOTYPE FOR STANDARD EXIT PROGRAM PARAMETERS                *
 *****************************************************************
D Parameters      PR                  EXTPGM('EXITTYPE1')
 * Copy in the parameter definition
 /COPY CS_SrcFile,CS_ExtTp1P

 *****************************************************************
 * STANDARD EXIT PROGRAM PARAMETERS                              *
 *****************************************************************

D Parameters      PI
 * Copy in the parameter definition
 /COPY CS_SrcFile,CS_ExtTp1P

 * Don't try to process unless the required user-definable parameter
 * has been passed

C                   IF          %parms < 9
C                             or %addr(iaUserParam) = *NULL
C                   RETURN
C                   ENDIF

 * Email the file using CoolSpools Email. The email address could be
 * determined, for example, by using some data extracted from the
 * spooled file (perhaps a customer number) to look up the appropriate
 * email address in the database (e.g. customer file). Here the
 * recipient details are hard-coded.

 * Populate the "from" structure - sent by Sales

C                   CLEAR                   CM_FromStruct
C                   EVAL      CM_FromEmail = 'sales@ariadnesoftware.co.uk'
C                   EVAL      CM_FromName = 'ariadne software'

 * Initialize array of "to" structures

C                   CLEAR                   aaTo

 * Populate first "to" structure - send to Support


C                   CLEAR                   CM_ToStruct
C                   EVAL      CM_ToEmail = 'support@ariadnesoftware.co.uk'
C                   EVAL      CM_ToName = 'Support'
C                   EVAL      CM_ToType = '*PRI'
C                   EVAL      aaTo(1) = CM_ToStruct

 * Populate second "to" structure - Cc: to Info

C                   CLEAR                   CM_ToStruct
C                   EVAL      CM_ToEmail = 'info@ariadnesoftware.co.uk '
C                   EVAL      CM_ToName = 'Info'
C                   EVAL      CM_ToType = '*CC'
C                   EVAL      aaTo(2) = CM_ToStruct
C                   EVAL      wuRecipients = 2

 * Initialize array of attachment structures

C                   CLEAR                   aaAttach

 * Populate first attachment structure - attach the file just created
```

```
C                   CLEAR                   CM_AttStruct
C                   IF        iaDir = *BLANKS
C                   EVAL      CM_Path = %trim(iaStmFile)
C                   ELSE
C                   EVAL      CM_Path = %trim(iaDir)
C                                       + '/'
C                                       + %trim(iaStmFile)
C                   ENDIF
C                   EVAL      CM_Method = '*ATTACH'
C                   EVAL      CM_Content = '*BINARY'
C                   EVAL      CM_CodePage = CM_STMF
C                   EVAL      aaAttach(1) = CM_AttStruct
C                   EVAL      wuAttachments = 1

 * Populate message structure

C                   CLEAR                   CM_MsgStruct
C                   EVAL      CM_MsgTxt  = 'The file is attached.'
C                   EVAL      CM_MsgType = '*BOTH'

 * Send the mail message

C                   CALLP(E)  CM_SndMsg('CM_F0100'          :
C                                       CM_FromStruct       :
C                                       wuRecipients        :
C                                       'CM_T0100'          :
C                                       aaTo                :
C                                       'Demo message'      :
C                                       'CM_M0100'          :
C                                       CM_MsgStruct        :
C                                       wuAttachments       :
C                                       'CM_A0100'          :
C                                       aaAttach            :
C                                       CM_NORMAL           :
C                                       CM_NO               :
C                                       waEmailId           :
C                                       CM_ErrStruct        )

C                   RETURN
```

Using the option list APIs to set the file name and passwords

In this next example, exit program EXITTYPE2 uses a “Type 2” (*TYPE2) parameter list.

It is intended to be called each time a new stream file is generated in order to set the file name to be used and the passwords to be applied to the new file. The exit program is defined to CVTSPLSTMF using:

EXITPGM(*LIBL/EXITTYPE2 *TYPE2

or, for one of the format specific commands (CVTSPLPDF etc.):

EXITPGM((*LIBL/EXITTYPE2 *TYPE2 *STMFEND))

It is assumed that the **EXITPGMPRM(*POS)** or **EXITPGMPRM(*KEY)** option has been used to extract information from the report which is then used to generate the stream file name and passwords.

```
     *****************************************************************
     *                                                               *
     * PROGRAM NAME: EXITTYPE2                                       *
     *                                                               *
     * DESCRIPTION : Example of a CoolSpools exit program in ILE RPG *
     *                                                               *
     *               N.B. Specify *TYPE2 for parameter type!!!       *
     *                                                               *
     *               Demonstrates the use of the bindable option list*
     *               APIs to set the stream file name and email      *
     *               recipients. The color option is updated simply  *
     *               to demonstrate the use of the OptRtvItem and    *
     *               OptUpdItem APIs.                                *
     *                                                               *
     *               Create module using CRTRPGMOD                   *
     *                                                               *
     *               Create program using CRTPGM...                  *
     *                                    BNDSRVPGM(CS_SRVPGM)       *
     *                                    ACTGRP(CALLER)             *
     *****************************************************************

     /COPY CS_SrcFile,AR_OPTFNCP

     /COPY CS_SrcFile,CS_CvtAPID

     *****************************************************************
     * PROTOTYPE FOR STANDARD EXIT PROGRAM PARAMETERS                *
     *****************************************************************
    D Parameters      PR                  EXTPGM('EXITTYPE2')
     * Copy in the parameter definition
     /COPY CS_SrcFile,CS_ExtTp2P

    D wiRtnCode       S             10I 0
    D waFormat        S              8A
    D wuOptionId      S             10U 0
    D waScope         S              1A
    D wuFrom          S             10U 0
    D wuTo            S             10U 0
    D waContext       S             10A
    D wuOffset        S             10U 0
    D wuReturned      S             10U 0
    D wuAvailable     S             10U 0

     *****************************************************************
     * STANDARD EXIT PROGRAM PARAMETERS                              *
     *****************************************************************
```

```
D Parameters      PI
 * Copy in the parameter definition
 /COPY CS_SrcFile,CS_ExtTp2P

D luAvailable     S             10U 0
D luOffset        S             10U 0
D luLength        S             10U 0
D luScopeFrom     S             10U 0
D luScopeTo       S             10U 0
D luOptionId      S             10U 0
D laFormat        S              8A
D laPriority      S              1A
D laContext       S             10A

 * Don't try to process unless the required number of user-definable
 * parameters has been passed

C                   IF        iiParmCount < 3
C                   RETURN
C                   ENDIF

 * Add a password option structure scoped to the appropriate page range
 * in order to set the passwords and security for the new PDF

C                   CLEAR                   CS_PWD01
C                   EVAL      saPwd01Option = '*YES'
C                   EVAL      svPwd01User = 'USER_'
C                                         + %trim(ivUserParam001)
C                   EVAL      svPwd01Owner = 'OWNER_'
C                                         + %trim(ivUserParam001)
C                   EVAL      saPwd01AlwPrt = '*YES'
C                   EVAL      saPwd01AlwChg = '*NO'
C                   EVAL      saPwd01AlwCpy = '*YES'
C                   EVAL      saPwd01AlwNot = '*NO'
C                   EVAL      wiRtnCode = OptAddItem('CS_PWD01'     :
C                                                    %size(CS_PWD01):
C                                                    CS_PWD01       :
C                                                    OPT_PTY_PRI    :
C                                                    iiFirstPage    :
C                                                    iiLastPage     )
C                   IF        wiRtnCode <> OPT_OK
 * Insert error handling here
C                   ENDIF

 ********************************************************************
 * Add a stream file name structure scoped to the appropriate page
 * range in order to set actual name of the new PDF
 ********************************************************************
C                   CLEAR                   CS_STM01
C                   EVAL      saStm01Option = '*REPLACE'
C                   EVAL      svStm01Path = %trim(ivUserParam001)
C                                         + '.pdf'
C                   EVAL      wiRtnCode = OptAddItem('CS_STM01'     :
C                                                    %size(CS_STM01):
C                                                    CS_STM01       :
C                                                    OPT_PTY_PRI    :
C                                                    iiFirstPage    :
C                                                    iiLastPage     )
C                   IF        wiRtnCode <> OPT_OK
 * Insert error handling here
C                   ENDIF

C                   RETURN
```

Using the option list APIs to set the email options

In this next example, exit program EXITTYPE3 uses a “Type 3” (*TYPE3) parameter list.

It is intended to be called each time a new stream file is generated in order to set the email options (whether CoolSpools emails the file using CoolSpools Email) and the email recipients

It is defined to CVTSPLSTMF using:

EXITPGM(*LIBL/EXITTYPE3 *TYPE3

or, for one of the format specific commands (CVTSPLPDF etc.):

EXITPGM((*LIBL/EXITTYPE3 *TYPE3 *STMFEND))

It is assumed that the **EXITPGMPRM(*POS)** or **EXITPGMPRM(*KEY)** option has been used to extract information from the report which is then used to generate the stream file name and passwords.

```
     *****************************************************************
     *                                                               *
     * PROGRAM NAME: EXITTYPE3                                       *
     *                                                               *
     * DESCRIPTION : Example of a CoolSpools exit program  in ILE RPG *
     *                                                               *
     *               N.B. Specify *TYPE3 for parameter type!!!       *
     *                                                               *
     *               Demonstrates the use of the bindable option list *
     *               APIs to set the stream file name and email      *
     *               recipients. The color option is updated simply  *
     *               to demonstrate the use of the OptRtvItem and    *
     *               OptUpdItem APIs.                                *
     *                                                               *
     *               Create module using CRTRPGMOD                   *
     *                                                               *
     *               Create program using CRTPGM...                  *
     *                                    BNDSRVPGM(CS_SRVPGM)       *
     *                                    ACTGRP(CALLER)             *
     *****************************************************************

     /COPY CS_SrcFile,AR_OPTFNCP

     /COPY CS_SrcFile,CS_CvtAPID

     *****************************************************************
     * PROTOTYPE FOR STANDARD EXIT PROGRAM PARAMETERS                *
     *****************************************************************
    D Parameters      PR                  EXTPGM('EXITTYPE3')
     * Copy in the parameter definition
     /COPY CS_SrcFile,CS_ExtTp3P

    D wiRtnCode       S             10I 0
    D waFormat        S              8A
    D wuOptionId      S             10U 0
    D waScope         S              1A
    D wuFrom          S             10U 0
    D wuTo            S             10U 0
    D waContext       S             10A
    D wuOffset        S             10U 0
    D wuReturned      S             10U 0
    D wuAvailable     S             10U 0

     *****************************************************************
```

```
 * STANDARD EXIT PROGRAM PARAMETERS                                *
 ******************************************************************

D Parameters      PI
 * Copy in the parameter definition
 /COPY CS_SrcFile,CS_ExtTp3P

D luAvailable     S             10U 0
D luOffset        S             10U 0
D luLength        S             10U 0
D luScopeFrom     S             10U 0
D luScopeTo       S             10U 0
D luOptionId      S             10U 0
D laFormat        S              8A
D laPriority      S              1A
D laContext       S             10A

 * *TYPE3 exit programs are called at the end of processing with the
 * exit point field set to *END in order to allow them to do
 * housekeeping such as closing files

C                   IF        iaExitPoint = '*END'
C                   EVAL      *INLR = *ON
C                   RETURN
C                   ENDIF

 * Don't try to process unless the required number of user-definable
 * parameters has been passed

C                   IF        iiParmCount < 3
C                   RETURN
C                   ENDIF

 * Add a password option structure scoped to the appropriate page range
 * in order to set the passwords and security for the new PDF

C                   CLEAR                   CS_PWD01
C                   EVAL      saPwd01Option = '*YES'
C                   EVAL      svPwd01User = 'USER_'
C                                         + %trim(ivUserParam001)
C                   EVAL      svPwd01Owner = 'OWNER_'
C                                         + %trim(ivUserParam001)
C                   EVAL      saPwd01AlwPrt = '*YES'
C                   EVAL      saPwd01AlwChg = '*NO'
C                   EVAL      saPwd01AlwCpy = '*YES'
C                   EVAL      saPwd01AlwNot = '*NO'
C                   EVAL      wiRtnCode = OptAddItem('CS_PWD01'     :
C                                                    %size(CS_PWD01):
C                                                    CS_PWD01       :
C                                                    OPT_PTY_PRI    :
C                                                    iiFirstPage    :
C                                                    iiLastPage     )
C                   IF        wiRtnCode <> OPT_OK
 * Insert error handling here
C                   ENDIF

 ******************************************************************
 * Add a stream file name structure scoped to the appropriate page
 * range in order to set actual name of the new PDF
 ******************************************************************
C                   CLEAR                   CS_STM01
C                   EVAL      saStm01Option = '*REPLACE'
C                   EVAL      svStm01Path = %trim(ivUserParam001)
C                                         + '.pdf'
C                   EVAL      wiRtnCode = OptAddItem('CS_STM01'     :
C                                                    %size(CS_STM01):
C                                                    CS_STM01       :
```

```
C                                                   OPT_PTY_PRI    :
C                                                   iiFirstPage    :
C                                                   iiLastPage     )
C                   IF        wiRtnCode <> OPT_OK
 * Insert error handling here
C                   ENDIF

 ********************************************************************
 * Add an email recipient structure scoped to the appropriate page
 * range in order to set the details of the person to whom this file
 * will be emailed
 ********************************************************************
C                   CLEAR                   CS_EMT01
C                   EVAL      svEMT01Email = %trim(ivUserParam001)
C                                          + '@ariadnesoftware.co.uk'
C                   EVAL      svEMT01Name = %trim(ivUserParam001)
C                                         + ' - '
C                                         + %trim(ivUserParam002)
C                                         + ' - '
C                                         + %trim(ivUserParam003)
C                   EVAL      saEMT01Type = '*PRI'
C                   EVAL      wiRtnCode = OptAddItem('CS_EMT01'     :
C                                                    %size(CS_EMT01):
C                                                    CS_EMT01       :
C                                                    OPT_PTY_PRI    :
C                                                    iiFirstPage    :
C                                                    iiLastPage     )
C                   IF        wiRtnCode <> OPT_OK
 * Insert error handling here
C                   ENDIF

 ********************************************************************
 * Update the colour option
 ********************************************************************
C                   EVAL      wiRtnCode = OptRtvItem('CS_CLR01'     :
C                                                    OPT_LAST       :
C                                                    luOptionId     :
C                                                    laFormat       :
C                                                    laPriority     :
C                                                    luScopeFrom    :
C                                                    luScopeTo      :
C                                                    laContext      :
C                                                    luOffset       :
C                                                    %size(CS_CLR01):
C                                                    CS_CLR01       :
C                                                    luLength       :
C                                                    luAvailable    )
C                   IF        wiRtnCode = OPT_OK
C                   EVAL      saClr01Color = '*BLUE'
C                   EVAL      saClr01BckClr = '*PALEYLW'
C                   EVAL      wiRtnCode = OptUpdItem(luOptionId     :
C                                                    'CS_CLR01'     :
C                                                    %size(CS_CLR01):
C                                                    CS_CLR01       :
C                                                    OPT_PTY_PRI    :
C                                                    iiFirstPage    :
C                                                    iiLastPage     )
C                   ENDIF

C                   RETURN
```

Renaming the stream file just created

In this next example, exit program EXITPGMRNM uses a “Type 1” (*TYPE1) parameter list.

It is intended to be called each time a new stream file is generated in order to rename the stream file just created to something more meaningful. Please note that if you use an *STMFSTR exit program to generate a CS_STM01 option structure, you can now override the stream file name before the file is created, making this renaming of stream files redundant.

It is defined to CoolSpools using the following syntax for the EXITPGM parameter of the CVTSPLSTMF command:

EXITPGM(*LIBL/EXITPGMRNM *TYPE1)

or, for one of the format specific commands (CVTSPLPDF etc.):

EXITPGM((*LIBL/EXITPGMRNM *TYPE1 *STMFEND))

It is assumed that the **EXITPGMPRM(*POS)** or **EXITPGMPRM(*KEY)** option has been used to extract an item of information from the report so that it is passed to this exit program in the ninth parameter position. This could be, for example, a customer number, order number, invoice number etc. The stream file is renamed from the name given it by CVTSPLSTMF to nnnnnnn.pdf, where nnnnnnn is the number extracted from the report and passed as a parameter.

```
 ****************************************************************
 *                                                              *
 * PROGRAM NAME: EXITPGMRNM                                     *
 *                                                              *
 * LANGUAGE    : ILE RPG                                        *
 *                                                              *
 * DESCRIPTION : Sample CVTSPLSTMF exit program                 *
 *                                                              *
 ****************************************************************

H COPYRIGHT('(C) @riadne software July 2000')

 ****************************************************************
 * PROTOTYPE FOR STANDARD EXIT PROGRAM PARAMETERS               *
 ****************************************************************

D Parameters      PR                  ExtPgm('EXITPGMRNM')

 * Spooled file name

D   iaSplFile                   10A

 * Name of the job which created the spooled file

D   iaSplJob                    10A

 * User id of the job which created the spooled file

D   iaSplUser                   10A

 * Job number of the job which created the spooled file

D   iaSplJobNo                   6A

 * Spooled file number of the spooled file
```

```
D   iiSplNbr                    10I 0

 * Spooled file user data

D   iaUsrDta                    10A

 * Name of the stream file created

D   iaStmFile                  128A

 * IFS directory in which stream file was created

D   iaDir                      256A

 * Exit program parameter string extracted from spooled file

D   iaExitParm                1024A

 ****************************************************************
 * PROTOTYPE FOR QCMDEXC                                        *
 ****************************************************************

D Command         PR                  ExtPgm('QCMDEXC')
D   iaCommand                32767A   CONST OPTIONS(*VARSIZE)
D   inCmdLen                   15P 5 CONST

 ****************************************************************
 * STANDARD EXIT PROGRAM PARAMETERS                             *
 ****************************************************************

D Parameters      PI
D   iaSplFile                   10A
D   iaSplJob                    10A
D   iaSplUser                   10A
D   iaSplJobNo                   6A
D   iiSplNbr                    10I 0
D   iaUsrDta                    10A
D   iaStmFile                  128A
D   iaDir                      256A
D   iaExitParm                1024A

D laOldName       S           1024A
D laNewName       S           1024A

D QUOTE           C                   ''''

 * If customer number found and passed as parameter, rename stream file

C                   IF        iaExitParm <> *BLANKS

 * Set up full path name to stream file

C                   IF        iaDir = *BLANKS
C                   EVAL      laOldName = QUOTE
C                                       + %trim(iaStmFile)
C                                       + QUOTE
C                   ELSE
C                   EVAL      laOldName = QUOTE
C                                       + %trim(iaDir)
C                                       + '/'
C                                       + %trim(iaStmFile)
C                                       + QUOTE
C                   ENDIF

 * Set up new path name for file using customer number in parameter

C                   EVAL      laNewName = QUOTE
```

```
C                                       + %trim(%subst(iaExitParm:1:7) )
C                                       + '.pdf'
C                                       + QUOTE

 * Rename the stream file

C                   CALLP(E)   Command(   'REN '
C                                       + laOldName
C                                       + ' '
C                                       + laNewName:2053)

C                   ENDIF

C                   RETURN
```

Emailing the stream file just created using SNDDST

In this next example, exit program EXITSNDDST uses a “Type 2” (*TYPE2) parameter list.

It is intended to be called each time a new stream file is generated in order to e-mail the stream file just created to the customer to whom the document relates using IBM’s SNDDST command.

EXITPGM(*LIBL/EXITSNDDST *TYPE2)

or, for one of the format specific commands (CVTSPLPDF etc.):

EXITPGM((*LIBL/EXITSNDDST *TYPE2 *STMFEND))

It is assumed that the **EXITPGMPRM(*POS)** or **EXITPGMPRM(*KEY)** option has been used to extract the customer number from the report. This customer number is used to look up the customer’s e-mail address on the customer database. The IBM I (OS/400) SNDDST command is then used to e-mail the stream file to the customer concerned.

Please note that SNDDST can only e-mail documents stored in the QDLS file system (“shared folders”). If you wish to email documents held outside of QDLS, please consider using ariadne’s CoolSpools Email product instead (see www.ariadnesoftware.co.uk/CoolSpools Email.htm).

```
 ****************************************************************
 *                                                              *
 * PROGRAM NAME: EXITSNDDST                                     *
 *                                                              *
 * APPLICATION : CVTSPLSTMF command                             *
 *                                                              *
 * LANGUAGE    : ILE RPG                                        *
 *                                                              *
 * AUTHOR      : Peter Clifford                                 *
 *                                                              *
 * DATE WRITTEN: Sep. 2000                                      *
 *                                                              *
 * DESCRIPTION : Sample CVTSPLSTMF exit program for sending a   *
 *               stream file via e-mail using SNDDST.           *
 *                                                              *
 ****************************************************************

H COPYRIGHT('(C) @riadne software July 2002')

 * Customer file

FCustFile  IF   E           K DISK
```

```
 ****************************************************************
 * PROTOTYPE FOR STANDARD EXIT PROGRAM PARAMETERS               *
 ****************************************************************

D Parameters      PR                  EXTPGM('EXITSNDDST')

 * Spooled file name

D   iaSplFile                   10A

 * Name of the job which created the spooled file

D   iaSplJob                    10A

 * User id of the job which created the spooled file

D   iaSplUser                   10A

 * Job number of the job which created the spooled file

D   iaSplJobNo                   6A

 * Spooled file number of the spooled file

D   iiSplNbr                    10I 0

 * Spooled file user data

D   iaUsrDta                    10A

 * Name of the stream file created

D   iaStmFile                  128A

 * IFS directory in which stream file was created

D   iaDir                      256A

 * Page number of the first page in the range processed

D   iiFirstPage                 10I 0

 * Page number of the last page in the range processed

D   iiLastPage                  10I 0

 * Exit program parameter count

D   iiParmCount                 10I 0

 * User-definable exit program parameter strings. Up to 150 of these
 * may follow. Define as many as you need.

D   iaUserParam1                999A         OPTIONS(*NOPASS)
D                                            VARYING

D   iaUserParam2                999A         OPTIONS(*NOPASS)
D                                            VARYING

D   iaUserParam3                999A         OPTIONS(*NOPASS)
D                                            VARYING

 ****************************************************************
 * PROTOTYPE FOR QCMDEXC                                        *
 ****************************************************************

D Command         PR                  ExtPgm('QCMDEXC')
```

```
D   iaCommand                32767A   CONST OPTIONS(*VARSIZE)
D   inCmdLen                   15P 5 CONST

 ****************************************************************
 * STANDARD EXIT PROGRAM PARAMETERS                             *
 ****************************************************************

D Parameters      PI
D   iaSplFile                 10A
D   iaSplJob                  10A
D   iaSplUser                 10A
D   iaSplJobNo                6A
D   iiSplNbr                  10I 0
D   iaUsrDta                  10A
D   iaStmFile                128A
D   iaDir                    256A
D   iiFirstPage               10I 0
D   iiLastPage                10I 0
D   iiParmCount               10I 0
D   iaUserParam1              999A         OPTIONS(*NOPASS)
D                                          VARYING
D   iaUserParam2              999A         OPTIONS(*NOPASS)
D                                          VARYING
D   iaUserParam3              999A         OPTIONS(*NOPASS)
D                                          VARYING

D waCustomerNo    S            10A
D waCommand       S          1024A
D wnCmdLen        S            15P 5

D QUOTE           C                   ''''

 * This example assumes that the EXITPGMPRM, EXITPGMPOS and EXITPGMKEY
 * parameters have been used to extract the customer number from the
 * spooled file, and that the first user-definable parameter contains the
 * customer number (assumed to be 10 characters for the purposes of
 * this example).

 * Make sure at least one parameter was passed and that the parameter
 * passed is not to short

C                   IF          iiParmCount < 1
C                            or %len(iaUserParam1) < %size(waCustomerNo)
 * Insert appropriate error handling here
C                   RETURN
C                   ENDIF

 * Extract the customer number from the parameter string

C                   EVAL      waCustomerNo = %subst(iaUserParam1      :
C                                                   1                 :
C                                                   %len(waCustomerNo))

 * Retrieve the customer's e-mail address from the customer file

C     waCustomerNo  CHAIN     CustFile
C                   IF        not %found(CustFile)
 * Insert appropriate error handling here
C                   RETURN
C                   ENDIF

 * Build the SNDDST command
C                   EVAL      waCommand = 'SNDDST TYPE(*DOC) '
C                                       + 'TOINTNET(('
 * "CustEMail" is assumed to be the customer's email address field
 * from the customer file
C                                       + %trim(CustEMail)
```

```
C                                       + ')) '
 * DSTD parameter below is the "subject" line that will appear for
 * the e-mail. Substitute your choice of subject line
C                                       + 'DSTD('
C                                       + QUOTE
C                                       + 'Your invoice from CoolSpools'
C                                       + QUOTE
C                                       + ') '
 * MSG parameter below is a brief message to accompany the attachment.
 * Substitute your choice of message.
C                                       + 'MSG('
C                                       + QUOTE
C                                       + 'Your invoice is'
C                                       + ' attached. Thank you for your '
C                                       + 'custom.'
C                                       + QUOTE
C                                       + ') '
 * The document name and folder names are passed into this program from
 * CVTSPLSTMF. The document name if generated from the TOSTMF parameter
 * with the addition of a numeric suffix for each spooled file created.
C                                       + 'DOC('
C                                       + %trim(iaStmFile)
C                                       + ') '
C                                       + 'FLR('
C                                       + %trim(iaDir)
C                                       + ')'

 * E-mail the stream file to the customer
C                   CALLP(E)  Command(waCommand:wnCmdLen)

C                   IF        %error
 * Insert appropriate error handling here
C                   ELSE
 * Probably a good idea to log the sending of the e-mail in some way
 * e.g. for CRM purposes. Also consider requesting confirmation of
 * delivery for CFMDEL(*YES)
C                   ENDIF

C                   RETURN
```

CoolSpools Environment Variables

Environment variables are an IBM i (IBM I (OS/400)) operating system feature that allows a series of key-value pairs to be defined at two levels:

- **system level**
- **job level**

OS/400 (IBM i) provides the following commands are available to help you manage environment variables:

- **ADDENVVAR** (add a new environment variable)
- **CHGENVVAR** (change the value of an existing environment variable)
- **RMVENVVAR** (remove i.e. delete an new environment variable)
- **WRKENVVAR** (work with a list of existing environment variables)

In addition, CoolSpools provides the following commands, intended primarily to assist with migrating your environment variables from one system or one partition to another:

- **SAVENVVAR** (save the current values of system-level environment variables into a stream file)
- **RSTENVVAR** (restore the values of system-level environment variables from a stream file)

Each job has its own set of environment variables. These are normally initialized from the system-level environment variables when the job starts. In the case of a batch job, the job-level environment variables may be copied from the submitting job's job-level environment variables if

SBMJOB ... CPYENVVAR(*YES)

is specified.

Please note that, if you set an environment variable at *SYS level, that environment variable will not take effect for jobs that are already running. You will need to restart CoolSpools server jobs (e.g. those that run in the COOLSPOOLS subsystem and CoolSpools SMTP servers) for your changes to take effect.

Also please note that environment variable names and their values are both case-sensitive.

CoolSpools uses environment variables to control a number of aspects of its processing. In many cases, an environment variable can be used to modify the default way in which CoolSpools behaves. These environment variables are documented below.

IBM environment variables used by CoolSpools

Name	Description	Default value	Other possible values
QIBM_AFP_RESOURCES_PATH	Path in which to look for AFP resources specified by the DDS AFPRSC keyword where no other path is specified.	/QIBM/UserData/OS400/ AFPresources	Path name
QIBM_NOTIFY_CRTSPLF	Name of a data queue on which an entry is placed every time a spooled file is created. Used by CoolSpools Spool Admin if a generic name or *ALL is specified for the output queue to monitor.	None	*DTAQ lib_name/dtaq_name

General

Name	Description	Default value	Other possible values
AR_ERR_CREATE_LOG_FILE	Whether a log file is created when a program error occurs.	*YES	*NO
AR_ERR_SEND_LOG_FILE	Whether the log file is sent to ariadne for diagnosis automatically when a program error occurs.	*YES	*NO
AR_MEM_CHK_DSK_STS	Whether to check for the available system memory (disk space) available when allocating storage. Setting this to 0 will potentially improve performance because no such checks will occur but means that you are potentially exposed to the system running out of disk space if storage used by CoolSpools exceeds the storage available.	1 (true)	0 (false)
AR_MEM_MIN_MEMORY	The minimum amount of system disk space that must be available for CoolSpools to continue processing, when CoolSpools is checking available storage.	16Mb	Specified in Mb, i.e. a value of 32 = 32Mb
AR_MEM_MAX_MEMORY	The maximum amount of storage CoolSpools is permitted to use.	4Gb	Specified in Mb, i.e. a value of 1024 = 1024Mb
AR_MEM_MAX_INCREMENT	The maximum amount by which CoolSpools is permitted to increase a memory pool at a single increment.	64 Mb	Specified in Mb, i.e. a value of 16 = 16Mb
AR_DISPLAY_USAGE_TIPS	Whether or not usage tips are shown	*LICENSED. Usage tips are shown if CoolSpools is not licensed, but not if it is licensed.	*YES Usage tips are always shown. *NO Usage tips are never shown.
CP_FTP_TIMEOUT	The default timeout used when doing FTP	60 seconds	Specify a value in seconds.
CP_FTP_PASSIVE	Whether passive FTP is used or not	0 (false, passive FTP not used))	1 (true, passive FTP used).
CP_FTP_LOGGING	Whether FTP commands are logged to the joblog or not	0 (false, no logging)	1 (true, commands are logged)

CoolSpools Spool Converter etc

Name	Description	Default value	Other possible values
CS_VAR_LEFT_MARKER	The marker identifying the start of a CoolSpools variable name	<:	Any combination of up to 10 characters.
CS_VAR_RIGHT_MARKER	The marker identifying the end of a CoolSpools variable name	:>	Any combination of up to 10 characters.
CS_FCN_MARKER	The marker identifying the start of a CoolSpools function name	$$	Any combination of up to 10 characters.
CS_XLT_TBL_ARABIC	Custom translation table for Arabic	None	LIBRARY/SRCFILE(MEMBER)
CS_MRG_METHOD	Whether or not a “soft” merge is permitted. A soft merge allows the use of the PDF Incremental Update feature to append one PDF to the end of another.	*SOFT PDF Incremental Updating is permitted.	*HARD Always force a “hard” merge, i.e. parse the input file in its entirety and re-generate.
CS_EXIT_PGM_TYPE4_UDP_FORMAT	The name of the format to be used for user-defined parameters with *TYPE4 exit programs.	CS_UDP01	CS_UDP02
CS_CRT_DIR_PATH	Whether or not directories in the TOSTMF path are created automatically if they do not already exist.	*NO Directories must already exist	*YES Directories in the path are created automatically if they do not already exist.
CS_RSC_DIR	The name of the directory in which CoolSpools will look for PCL resources (macros and soft fonts), if not otherwise specified	*TODIR CoolSpools will look in the directory into which the output file is being created	*CURDIR The current directory path Specify a path
CS_PCL_AUTO_MACROS	Whether PCL automatic macros are implement or not	*YES PCL automatic macros will be used.	*NO PCL automatic macros are inored.
CS_TXT_FF_CRLF	Whether the formfeed character written to text files when a formfeed is included at the end of a page should be followed by linefeed or not	*NO	*YES
CS_PDF_TOUNICODE	Whether or not fonts in PDF files should have a ToUnicodeCMap generated for them.	*YES	*NO
CS_DFT_PRT_DEV	The default printer device to assume. This is the device CoolSpools will emulate, by default.	*SYSVAL The device identified by the QPRTDEV system value	printer_device_name *HPT Generic Host Print Transform Printer *IPDS Generic IPDS printer
CS_QPRTDEV_MSG	What message, if any, is issued if the QPRTDEV system value does not refer t a valid printer device.	*NONE No message is issued	*ERROR An error message. Processing stops. *WARNING A warning message. Processing continues.
CS_PDF_DCP_GROUP_4	Whether or not to decompress images compressed with CCITT Group 4	*YES Images compressed with CCITT compression are decompressed and	*NO Images compressed with CCITT Group 4 compression are implemented inside a PDF

	compression	recompressed with PDF using the default PDF compression method.	file in their original compression.
CS_TXT_LINE_CALC	Provides backwards compatibility with Version 4 of CoolSpools in relation to	None	V4R1M0 Line numbers are calculated as they were inv CoolSpools V4.
CS_FNT_IPDS_USE_FONT_CPI	For an *IPDS spooled file, determines whether the CPI value is set from the FONT attribute of the spooled file or the CPI attribute.	*YES The FONT attribute overrides the CPI attribute and determines the CPI value used when converting to text etc.	*NO The CPI attribute is used.
CS_FNT_HPT_SCL	Determines whether the FONT(*SCALE) causes fonts in a spooled file to be modified (height: width ratio to be exaggerated) when a Host Print Transform printer device is selected.	*NO FONT(*SCALE) is not applied to fonts in spooled files when an HPT device is selected.	*YES FONT(*SCALE) is not applied to fonts in spooled files when an HPT device is selected.
CS_FNT_SCALE_IPDS	Determines whether the FONT(*SCALE) causes fonts in an IPDS spooled file to be modified (height: width ratio to be exaggerated) or not.	*NO FONT(*SCALE) is not applied to fonts in IPDS spooled files.	*YES FONT(*SCALE) is applied to fonts in IPDS spooled files.
CS_FNT_SCALE_AFPDS	Determines whether the FONT(*SCALE) causes fonts in an AFPDS spooled file to be modified (height: width ratio to be exaggerated) or not.	*NO FONT(*SCALE) is not applied to fonts in AFPDS spooled files.	*YES FONT(*SCALE) is applied to fonts in AFPDS spooled files.
CS_FNT_SCALE_NON_SCALABLE	When a font size is specified for a non-scalable font (e.g. DDS FONT(... (*POINTSIZE)) is used for a font number that implies a font size such as 222), this controls whether the font size specified is used or ignored.	*NO The font size specified is ignored and the font size implied by the font number is used.	*YES The specified font size is used in pace of the font size implied by the font number.
CS_TXT_LINE_METHOD	Controls the method used to calculate line numbers in spooled files other than *USERASCII.	*NEW Corrected method. Recommended unless backwards compatibility required.	*OLD Previous, erroneous method. Recommended only if backwards compatibility required.
CS_TXT_PCL_LINE_METHOD	Controls the method used to calculate line numbers in PCL spooled files.	*NEW Corrected method. Recommended unless backwards compatibility required.	*OLD Previous, erroneous method. Recommended only if backwards compatibility required.
CS_FNT_OUTLINE_FOR_RAST	When a font resource name is used (e.g. DDS FNTCHRSET or CDEFNT attributes) and that font resource specifies a raster font, whether the system will substitute an equivalent outline font if one is available. Outline fonts normally give better results in PDF that raster fonts.	*YES Use the equivalent outline font, if available.	*NO Use the original raster font.
CS_MSG_LEVEL	The level of messages output by CoolSpools. This can be used to	*NORMAL Normal level	*NONE No message are sent *MIN Minimal messages *MAX Maximum diagnostic

	increase or reduce the number of messages sent to the joblog. Not yet fully implemented.		information.
CS_PDF_DUPLEX_DUMMY_PAGES	Whether, when converting a duplex spooled file, an additional empty, “dummy” page, intended to act as a placeholder for the back side of the form, is written to the PDF to allow easy printing of the PDF in duplex mode.	*NO No additional pages are written.	*YES Where necessary, additional “dummy” pages are written to assist with duplex printing of PDFs.
CS_CTL_TRIM_SPACES	Whether, when converting a PCL spooled file, unnecessary spaces are removed from text items in order to minimize file size.	*NO Text is output including any space characters at the beginning or end.	*YES Spaces are removed from the beginning or end of the text and page coordinates adjusted, if necessary.
CS_FNT_MAP_WIDTHS	Whether, in order to calculate the space occupied by text on the page, a font widths table is generated.	*YES Use a font widths table. This has performance advantages and is now recommended.	*NO Do not use a font widths table. This should only be used if use of a font widths table gives problems.
CS_PDF_DCP_IBM_MMR	Whether images compressed using IBM’s MMR compression algorithm are full decompressed and re-compressed using the default PDF compression algorithm (flate).	*NO IBM MMR images are not fully decompressed. IBM MMR images can normally be handled by making minor modifications to transform them into CCITT compressed images which are compatible with PDF.	*YES Decompress IBM MMR images fully. This option should only be used if problems with an MMR compressed image are encountered.
CS_APY_PAG_RTT_MGN	Whether or not to use the new method of calculating page margins when a page rotation occurs.	*YES The new method is used.	*NO The old method is used. Use this value if problems are experienced with the new method or for reasons of backwards compatibility.
CS_APY_PAG_SEG_MGN	When the spooled file has the attribute FRONTMGN(*DEVD), which implies that the positioning of objects on the page should be determined based on the no-print border of the printer device on which the spooled file is to be printed, CoolSpools uses this setting to control whether or not to apply the margins it has assumed (based on the values supplied on the PRTDEV parameter) when calculating the positioning of page segments.	*NO Assumed margins are not applied. If a user-supplied margin setting has been specified on the PRTDEV parameter, this value will be used, but margin settings assumed based on the printer device nominated or inferred from the QPRTDEV system value will not be applied to page segments.	*YES Assumed or calculated page margins are applied to page segments.
CS_PDF_HPT_OVL_IMG	Whether, when a Host Print Transform printer device has been selected, images in overlays are output first, before any other items on the page, to ensure that text	*NO Images in overlays are not output first. This could result in the image overlaying text in some circumstances.	*YES The images are output first. This could correct the overlaying of images over text in some instances.

	overlays the images rather than the reverse.		
CS_PDF_NON_HPT_OVL_IMG	Whether, when a Host Print Transform printer device has been not been selected, images in overlays are output first, before any other items on the page, to ensure that text overlays the images rather than the reverse.	*NO Images in overlays are not output first. This could result in the image overlaying text in some circumstances.	*YES The images are output first. This could correct the overlaying of images over text in some instances.
CS_AFP_HPT_LIN_ADJ	Whether, when a Host Print Transform printer device is selected, page coordinates are adjusted to reflect the way in which HPT moves text out of the printer no-print border (unlike true IPDS printers).	*YES HPT adjustments to the position of text in no-print borders are implemented.	*NO HPT adjustments to the position of text in no-print borders are not implemented.
CS_EXIT_TYPE1_PASS_MISSING_PARM	When a *TYPE1 exit program is being called, and no user-defined command parameter is available (e.g. selects an empty area of the page or no key string found), whether the user-defined command parameter is passed or not.	*YES The parameter is passed as an empty string.	*NO The parameter is not passed at all.
CS_EXIT_TYPE23_PASS_MISSING_PARM	When a *TYPE2, *TYPE3 or *TYPE4 exit program is being called, and no user-defined command parameter is available (e.g. selects an empty area of the page or no key string found), whether the user-defined command parameter is passed or not.	*YES The parameter is passed as an empty string.	*NO The parameter is not passed at all.
CS_PDF_MRG_OVR_PRT	When an *SCS spooled file contains text items that overlap, whether those text items are output as separate overlapping text items inside the PDF, or merged into a single text item.	*NO Text items are not merged.	*YES Text items are merged. This can sometimes be necessary to ensure correct handling of bolding or underlining.
CS_STMFOPT_ADD_ERR	Whether an error occurs an processing stops if STMFOPT(*ADD) is specified and the target file does not already exist.	*NO No error occurs. A warning message is sent and the file is created.	*YES An error message is sent and processing stops without creating the file.
CS_PAGRTT_COR_RULES	Determines what rules are implemented in relation to page rotation and COR (Computer Output Reduction)	*NEW The new rules introduced in V6. These are believed to simulate the behavior of most printers more closely than previously was the case.	*OLD The old rules used in V5 and earlier. Use this option is the new rules produce unwanted results.
CS_AFP_PAGE_SIZE	When converting an AFP spooled file, and PAGESIZE(*CALC) is	The default is to use the country code of the job to determine the paper size:	*SPLF Always trust the page size attributes of the spooled file and use the paper size

	specified, this variable controls whether the paper size is determined by the country code or whether the spooled file attributes are used to determine the page size.	letter for US and CA, A4 elsewhere.	implied by them. *NOTDFT Where the page size attributes of the spooled file are not the defaults (132 columns by 66 lines), trust the page size attributes and use those values. *FORMTYPE Where the form type is not *STD, trust the page size attributes and use those values.
CS_PCL_PATTERN_METHOD	Whether patterns in PCL spooled files are implemented in PDF as a dot pattern or a grayscale shading.	*GRAYSCALE PCL patterns are converted to gray shading. This usually gives a better visual appearance that the use of a raster dot pattern.	*PATTERN A raster dot pattern is used.
CS_OPN_SPLF_ACTION	Determines how CoolSpools handles the situation where a spooled file is still open when it comes to process it. This can cause problems, especially in relation to *USERASCII spooled files, where CoolSpools could attempt to process an incomplete printer data stream, with unpredictable results.	*DEVTYPE If the spooled file is *USERASCII, an error message is issued and processing stops. For all other spooled file types, processing continues.	*IGNORE Processing continues irrespective of the spooled file type. *WARNING A warning message is issued but processing continues. *ERROR An error message is issued and processing stops. *WAIT. CoolSpools waits for the file to be closed. The number of seconds to wait can be defined on environment variable CS_OPN_SPLF_TIMEOUT and the poll interval between checks to see if the file is still open can be set on env var CS_OPN_SPLF_POLL_INTERVAL.
CS_OPN_SPLF_TIMEOUT	See CS_OPN_SPLF_ACTION above	600 seconds (10 minutes)	A value in seconds between 1 and 3600.
CS_OPN_SPLF_POLL_INTERVAL	See CS_OPN_SPLF_ACTION above	2 seconds	A value in seconds between 1 and 60
CS_POSTSCRIPT_ENCODING_xx	Defines a custom encoding scheme for a given script (xx = script code e.g. JA= Japanese Kanji, KA=Japanese katakana etc.)	None	LIBRARY/SRCFILE(MEMBER)
CS_MRG_PDF_API	The API to use when merging spooled files with the MRGPDF command or when implementing STMFOPT(*ADD) with PDF	CS_MRGAPIR The original merge API.	CS_PDFMRGR The new merge API (currently still experimental). This API will probably improve performance, especially where several files are being merged simultaneously.
CS_MRG_ALW_APPEND	Whether, when the new merge API CS_PDFMRGR is being used, the API should, where possible, append directly to an existing PDF file rather than write to a temporary file first.	*NO A temporary file is always used. The original file is not replaced until the temporary file has been built in its entirety. This is slightly less efficient and uses more working storage, but minimizes the risk of corrupting the original file in the event of a failure.	*YES Where possible, a temporary file is not used and the original file is directly appended to. This optimizes performance and minimizes the amount of working storage needed, but, in the event of a failure, the original file could be left in a corrupt state.
CS_EMAIL_SEND_ERROR_ACTION	What action CoolSpools Spool	*STOP Processing stops immediately. If, therefore,	*CONTINUE A warning message is issued but

	Converter takes if it is unable to send an email with an attachment, e.g. because no email address is available or an invalid (not well formed) email address is specified.	a spooled file is being split into multiple output files, and an email error occurs on one output file, any subsequent files will not be processed.	processing continues and any remaining files will be processed if they can be. *EMAIL:email The email will be sent to the email address specified after the colon instead. Processing then continues. *ADRL:list The email will be sent to the email address list specified after the colon instead. Processing then continues.
CS_EMAIL_LOG_ENCRYPTED	Whether, when an email is to be logged, it is logged in compressed form.	*NO The email is not encrypted when it is logged.	*YES The email is encrypted using the password specified on the CS_EMAIL_SAVE_PASSWORD variable.
CS_EMAIL_LOG_OPTION	Whether, by default, emails are logged for audit purposes in log files CM_MSGLOG etc.	*YES The email is logged	*NO The email is not logged
CS_EMAIL_METHOD	The default method by which emails are delivered	*MSF IBM's Mail Server Framework and the IBM SMTP server are used	*SMTP The CoolSpools SMTP server is used. This is now the recommended method.
CS_EMAIL_SAVE_COMPRESSED	Whether, when an email is to be saved, it is saved in compressed form.	*YES The email is saved in a compressed form.	*NO The email is saved in uncompressed form.
CS_EMAIL_SAVE_DAYS	When an email is saved, the default retention period (in days) to assign. The email becomes eligible for deletion using the DLTCMNMSG command with the DLTSAVMSG(*MSG) option after the specified number of days.	*NOMAX (no limit)	1-9999
CS_EMAIL_SAVE_ENCRYPTED	Whether, when an email is to be saved, it is saved in compressed form.	*NO The email is not encrypted when it is saved/	*YES The email is encrypted using the password specified o the CS_EMAIL_SAVE_PASSWORD variable.
CS_EMAIL_SAVE_OPTION	Whether, by default, emails are saved after being sent to allow them to be resent later using RSNCMNMSG.	*NO The email is not saved	*YES The email is saved
CS_EMAIL_SAVE_PASSWORD	The password to be used when CS_EMAIL_SAVE_ENCRYPTED is *YES	None.	The password to be used. This must be specified in the form of a hex string as returned by the DSPENCPWD command.
CS_TXT_LINE_NUMBERS	When converting to text format, whether or not line numbers are included.	*NO No line numbers are output.	*YES A line number is output at the start of each line.
CS_TXT_COLUMN_RULER	When converting to text format, whether or not a column ruler is included at the top of the page.	*NO No column ruler is output.	*YES A column ruler is output.
CS_XLS_LABELS_ONLY	When converting to Excel format, whether to output or data as labels (character cells) rather than numbers	*NO Output data as number cells where the data is numeric.	*YES Output all data as labels (character cells).

	(floats/integers).		
CS_DEV_MGN_UNIT	The units in which the margins defined below are specified	*INCH Inches	*CM Centimeters *MM Millimeters
CS_DEV_xxxxxxxxxx_LFT_MGN_PRT	The left margin to assume for device xxxxxxxxx when in portrait mode	Determined by the device model	Specify the margin in the units defined on CS_DEV_MGN_UNIT
CS_DEV_xxxxxxxxxx_RGT_MGN_PRT	The right margin to assume for device xxxxxxxxx when in portrait mode	Determined by the device model	Specify the margin in the units defined on CS_DEV_MGN_UNIT
CS_DEV_xxxxxxxxxx_TOP_MGN_PRT	The top margin to assume for device xxxxxxxxx when in portrait mode	Determined by the device model	Specify the margin in the units defined on CS_DEV_MGN_UNIT
CS_DEV_xxxxxxxxxx_BTM_MGN_PRT	The bottomt margin to assume for device xxxxxxxxx when in portrait mode	Determined by the device model	Specify the margin in the units defined on CS_DEV_MGN_UNIT
CS_DEV_xxxxxxxxxx_LFT_MGN_LND	The left margin to assume for device xxxxxxxxx when in landscape mode.	Determined by the device model	Specify the margin in the units defined on CS_DEV_MGN_UNIT
CS_DEV_xxxxxxxxxx_RGT_MGN_LND	The right margin to assume for device xxxxxxxxx when in landscape mode.	Determined by the device model	Specify the margin in the units defined on CS_DEV_MGN_UNIT
CS_DEV_xxxxxxxxxx_TOP_MGN_LND	The top margin to assume for device xxxxxxxxx when in landscape mode.	Determined by the device model	Specify the margin in the units defined on CS_DEV_MGN_UNIT
CS_DEV_xxxxxxxxxx_BTM_MGN_LND	The bottomt margin to assume for device xxxxxxxxx when in landscape mode.	Determined by the device model	Specify the margin in the units defined on CS_DEV_MGN_UNIT

CoolSpools Email

CM_ALLOW_A_GRAVE_FOR_AT	Whether or not an a-grave character (à) is acceptable as well as an @symbol in the email address. This provides compatibility with SNDDST in a French-language environment.	*YES Allow an à as well as an @	*NO An @ must be specified.
CM_VALID_EMAIL_CHARS	The characters that are permitted in an email address	ABCDEFGHIJKLMNOPQRSTUVWXYZabcedfghijklmnopqrstuvwxyz0123456789-_.+&'/=?	Specify the list of characters to be permitted.
CM_VALID_EMAIL_FIRST_CHARS	The characters that are to be permitted as the first character of an email address	ABCDEFGHIJKLMNOPQRSTUVWXYZabcedfghijklmnopqrstuvwxyz0123456789	Specify the list of characters to be permitted.
CM_VALID_EMAIL_NO_DOT	Whether or not an email address is valid if it contains no dot/period (.)	0 An email address must contain at least one dot/period (.)	1 An email address is valid without a dot/period (.)
CM_CHARSET_UTF8	Whether emails are sent, by default, in UTF8 encoding	1 UTF8 encoding is used by default.	0 Windows (1252) or some other appropriate Windows encoding is used.
CM_CONTENT_TYPE_PDF	The content type to be used for PDF attachments	application/pdf	Specify the content type e.g. application/octet-stream
CM_CONTENT_TYPE_XLS	The content type to be used for Excel (.xls) attachments	application/vnd.ms-excel	Specify the content type e.g. application/octet-stream
CM_CONTENT_TYPE_XLSX	The content type to be used for Excel (.xlsx) attachments	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet	Specify the content type e.g. application/octet-stream
CM_CONTENT_TYPE_HTML	The content type to be used for HTML attachments	text/html	Specify the content type
CM_CONTENT_TYPE_XML	The content type to be used for XML attachments	text/xml	Specify the content type
CM_CONTENT_TYPE_TEXT	The content type to be used for text attachments	text/plain	Specify the content type
CM_CONTENT_TYPE_CSV	The content type to be used for CSV attachments	text/plain	Specify the content type
CM_CONTENT_TYPE_JPG	The content type to be used for JPEG attachments	image/jpeg	Specify the content type
CM_CONTENT_TYPE_TIFF	The content type to be used for TIFF attachments	image/tiff	Specify the content type
CM_CONTENT_TYPE_RTF	The content type to be used for RTF attachments	application/rtf	Specify the content type
CM_CONTENT_TYPE_ZIP	The content type to be used for zip attachments	application/zip	Specify the content type

ST_CRT_TXN_ALL_SPLF	Whether spooled file transactions should be created for all spooled files, or just for those that match rules. This option can be used to minimize the number of spooled file transactions generated, especially when generic monitors are in use.	*NONGENERIC Transactions are created for all spooled files if the spooled file monitor is not generic (not *ALL or a generic* name), irrespective of whether the spooled file matches any rules or not. If no rule is matched to, there will be a transaction "header" with no rules steps to process. I the spooled file monitor is generic (*ALL or generic* name), a transaction is only crated if the spooled file matches one or more rules.	*NO A spooled file transactions is only crated if the spooled file in question matches at least one rule. *YES A spooled file transaction is created irrespective of whether the spooled file matches any rules or not.
ST_DTAQ_EXCL_LOCK	Whether or not an exclusive lock is allocated on the data queue associated with the spooled file monitor by the monitoring job.	*YES The data queue is locked by the monitor job.	*NO The data queue is not locked by the monitor job. use this option only if the lock causes problems.
ST_JOB_USR_PRF	The user profile under which Spool Admin server jobs in subsystem COOLSPOOLS run.	*CURRENT The user profile of the user that runs the STRMONSPLF command.	*JOBD The user profile associated with the COOLSPOOLS job description. user-profile Specify the user profile. You must ensure that for either option, the user ID has appropriate authorities.
ARIADNE__INCLUDE_SYSTEM_OUTPUT	Whether or not CoolSpools Spool Admin includes system output when WRKSPLFPDM is used. This is provided for compatibility with WRKSPLF, which does not show system output when ASTLVL(*BASIC) is specified. See next for details of what constitutes "system output".	*YES System output is shown.	*NO System output is not shown.
ARIADNE_SYSTEM_OUTPUT_SPLF_NAMES	What spooled file types are considered system output	QPJOBLOG;QPSRVDMP;QPPGMDMP	List the names of the spooled files to be considered system output, separated by semicolons

CoolSpools Database

CS_VAR_LEFT_MARKER	The marker identifying the start of a CoolSpools variable name	<:	Any combination of up to 10 characters.
CS_VAR_RIGHT_MARKER	The marker identifying the end of a CoolSpools variable name	:>	Any combination of up to 10 characters.
SL_FCN_MARKER	The marker identifying the start of a CoolSpools Database function name	$$	Any combination of up to 10 characters.
SL_DBFDATFMT	The date format to assume when CVTDBFxxx DBFDATFMT(*ENVVAR) is specified. This is the format assumed for numeric fields that are identified as storing dates based on the edit code or edit word associated with them.	*YMD YYMMDD	*DMY *MDY *CYMD *CDMY *CMDY
SL_CRT_DIR_PATH	Whether or not directories in the TOSTMF path are created automatically if they do not already exist.	*NO Directories must already exist	*YES Directories in the path are created automatically if they do not already exist.