FTPC118 and FTPC119

Combined SMF Exit and Print Programs for Type 118 and 119 FTP SMF Records

(Updated on June 5, 2005)

Paul Wells
Saudi Aramco Box 12959
Dhahran 31311
Saudi Arabia
Paul.Wells@aramco.com

Table of Contents

Preface

This documents describes two programs (FTPC118 and FTPC119) written in the C language to process SMF type 118/119 records from the FTP component of TCPIP on IBM's OS/390 and z/OS operating systems.

The programs operate in two modes: SMF exit mode and TEST mode.

In SMF exit mode the programs are installed as operating system SMF exits and invoked by the system each time an SMF record is written. In this mode the programs select FTP SMF records and issue formatted WTO (Write to Operator) messages for each FTP related record.

In TEST mode the programs run in batch against an input file of SMF data and print FTP related records to a file.

The primary motivation for the creation of these programs was to enable an installation to log FTP activity in real-time for tracking, audit and automation purposes and also to report on historical FTP activity. The secondary motivation was to show that the C language could be used for operating system exits on OS/390.

The author appreciates feedback as to good uses to which this material is being put, and would give due consideration to any suggestions for corrections, clarifications or enhancements. The author can be contacted as follows.

Name: *Paul Wells*
Address: *Saudi Aramco Box 12959*
Dhahran 31311
Saudi Arabia
Telephone: *+966 3 873 3155 (Work - direct line)*
Fax: *+966 3 873 8958 (Work)*
Email: *Paul.Wells@aramco.com*

Chapter 1—Installation

Upload TSO XMIT Dataset

The program source, object modules and associated sample JCL are supplied in a single installation dataset which is in TSO XMIT format. Most commonly this file would be obtained from FILE600 of the CBT Tape, typically downloaded from the CBT tape website. It would then be uploaded to TSO into a FB80 sequential dataset and the TSO RECEIVE command used to reload the installation dataset into a PDS. The installation PDS will usually be named `(userid).CBTxxx.FILE600.PDS`, once unloaded, where `xxx` is the generation number of the CBT tape. Its contents are as follows.

`$$README` — Quick start guide.
`$CHANGES` — Summary of changes.
`$CONTACT` — Contact details for the author.
`$DOCFTPC` — This document in PDF format.
`$INDEX` — Index of members.
`FTPDOC` — JCL to FTP documentation to workstation.
`PACKAGEJ` — JCL to package the installation dataset.
`RECV` — REXX EXEC to receive the three TSO XMIT format datasets.
`XMITPDSA` — Assembler source and associated sample JCL in TSO XMIT format.
`XMITPDSC` — C language source and associated sample JCL in TSO XMIT format.
`XMITPDSO` — Object modules in TSO XMIT format.
`XMITPDSR` — REXX language source and associated sample JCL in TSO XMIT format.

Get Documentation

The documentation (this document) is contained in member `$DOCFTPC` of the installation PDS. It is in PDF format and should be downloaded to a workstation in binary format and read with Adobe Acrobat Reader. A sample job named `FTPDOC`, is provided in the installation PDS to FTP the documentation to a workstation. If you don't have an FTP server on your workstation, I recommend War FTP Daemon.

Receive PDS Datasets

Execute the `RECV` member of the installation PDS to create the four constituent PDS datasets via TSO RECEIVE. The datasets will be named as follows.

`(userid).CBT.FILE600.ASM.PDS`—An FB80 dataset containing assembler source and associated sample JCL.

`(userid).CBT.FILE600.C.PDS`—A VB255 dataset containing C source and associated sample JCL.

`(userid).CBT.FILE600.OBJ.PDS`—An FB80 dataset containing object modules for those who cannot or do not wish to assemble/recompile the source programs.

(userid).CBT.FILE600.REXX.PDS— An FB80 dataset containing REXX source and associated sample JCL. (The REXX material is part of the CBT tape submission but unrelated to FTPC118/119)

RECV also allocates a load library used later for link-edits named (userid).CBT.FILE600.LOAD.

Enable SMF Type 118 and/or 119 Records

Program FTPC118 processes SMF type 118 records from the FTP component of TCPIP on OS/390 version 2.5 and upwards. Program FTPC119 processes SMF type 119 records from the FTP component of TCPIP on z/OS version 1.2 and upwards. It is recommended that type 119 records are used in preference to type 118, if they are available on your system. Type 119 records overcome some problems with type 118 records, are better structured and will be maintained and enhanced by IBM in future z/OS releases. Both programs can run side by side on the same system if both SMF type 118 and 119 records are enabled.

To enable client type 118 SMF records in TCPIP include the following statement in SYS1.TCPPARMS(PROFILE) or local equivalent.

- SMFCONFIG FTPCLIENT

If this statement needs to be changed an obey file can be used with the VARY TCPIP,,OBEY command.

To enable server type 118 SMF records in TCPIP include the following statement in SYS1.TCPPARMS(FTPSDATA) or local equivalent.

- SMF STD; SMF records use standard subtypes
- SMFJES; SMF recording when filetype=jes
- SMFSQL; SMF recording when filetype=sql

SMFJES and SMFSQL are optional but the SMF statement must specify STD as non-standard subtypes are not supported. The FTP server should be recycled to change this option.

To enable client type 119 SMF records in TCPIP (z/OS 1.2 and above only) include the following statement in SYS1.TCPPARMS(PROFILE) or local equivalent.

- SMFCONFIG TYPE119 FTPCLIENT

If this statement needs to be changed an obey file can be used with the VARY TCPIP,,OBEY command.

To enable server type 119 SMF records in TCPIP (z/OS 1.2 and above only) include the following statement in SYS1.TCPPARMS(FTPSDATA) or local equivalent.

- SMF TYPE119; SMF records (type 119)
- SMFJES TYPE119; SMF filetype=jes (type 119)
- SMFSQL TYPE119; SMF filetype=sql (type 119)

SMFJES and SMFSQL are optional. Both type 118 and 119 records can be defined on the same system if required. The FTP server should be recycled to change this option.

Check SYS1.PARMLIB(SMFPRMxx) or local equivalent to ensure that type 118/119 records are written and check your SMF dump jobs to verify that the records are collected and archived.

Run an SMF Extract

Optionally run an SMF extract of type 118 or 119 records using members `SMFX118J` or `SMFX119J` from `(userid).CBT.FILE600.C.PDS`. These jobs run sort to extract the FTP subtypes only. Examine the output the verify that some records were extracted. If not, refer back to *Enable SMF Type 118 and/or 119 Records*.

Link-Edit FTPC118/119

Run the link-edit for FTPC118 or FTPC119 from `(userid).CBT.FILE600.C.PDS` members `FTPC118L` or `FTPC119L`.

There are two steps in each link-edit. The first link-edits the TEST mode version of FTPC118/119 which creates load modules FTPT118/119 used in batch to process an SMF input dataset. The second link-edit step includes Systems Programming C (SPC) support and creates load modules FTPC118/119 used for the SMF exit mode of operation. FTPC118 has an alias of FTPSMFEX. This alternate entry point is described in *Appendix C—FTPSMFEX*.

Run Program in TEST mode

Run either the FTPT118 or FTPT119 load module against the SMF extract dataset created above. Sample JCL is in members `FTPT118J` or `FTPT119J` from `(userid).CBT.FILE600.C.PDS`. If the *Run an SMF Extract* step was skipped and the programs are running against a complete SMF tape, it is recommended that the SYSOUT DD statement is dummied out to prevent non-FTP SMF records being flagged.

FTPT119 has a program switch (-r) which can be set to enable TCP/IP domain name resolution of the IP addresses in the input SMF data. If enabled, the resolved domain names are included with the IP addresses in the printed output.

The job should complete with RC=0 and produce output to the SYSPRINT DD. Sample output can be seen in *Appendix A—Sample Output*. The output fields are described in *Appendix B—Explanation of Output Fields*.

Run SMF Exits Programs via FTPCTST

FTPCTST is a program to verify that the SMF exit modules are executable and formatting the SMF data correctly. It does this without the need to install FTPC118/119 as SMF exits. Installing and running this program is optional, but recommended.

Assemble and link-edit FTPCTST using member `ASMLKED` of `(userid).CBT.FILE600.ASM.PDS`. The SYSLMOD should output to the library created in *Receive PDS Datasets*.

Temporarily APF authorize the load library used above. This step is optional, but recommended, since the behavior of WTO is subtly different when APF authorized. If not APF authorized, the WTO lines from FTPC118/119 are one character shorter, and due to spacing adjustment, the output may not exactly match that which is achieved when running in an SMF exit.

Run FTPCTST using member FTPCTSTJ of (userid).CBT.FILE600.ASM.PDS. The parameter passed is the name of the SMF exit being tested i.e. FTPC118 or FTPC119. The SMFDD statement should point to the SMF dataset created in *Run an SMF Extract*.

If all is well, the job should complete RC=0 and produce output similar to that in *FTPC118 Output in SMF Exit Mode—JES2 Joblog/SYSLOG* or *FTPC119 Output in SMF Exit Mode—JES2 Joblog/SYSLOG*.

Remove the temporary APF authorization for the load library used above, if desired.

Copy SMF Exit Programs to a Linklist Library

The load modules FTPC118/119 should be copied from the load library used in *Link-Edit FTPC118/119* to an APF authorized system linklist library. FTPC118 has an alias of FTPSMFEX which is described in *Appendix C—FTPSMFEX*. If installing FTPC118 as an operating system SMF exit, then FTPSMFEX does need to be copied to the linklist.

The module(s) should be made accessible via an LLA refresh or equivalent, if necessary.

Enable IEFU83/IEFU84 SMF Exits

Ensure that exits IEFU83 and IEFU84 are being called for all address spaces by coding them in all SYS or SUBSYS statements in SYS1.PARMLIB(SMFPRMxx) e.g.

```
SYS(NOTYPE(..,..,..,..),
    EXITS(IEFU83,IEFU84,.....,.....,),.....,.....)
SUBSYS(STC,NOTYPE(..,..,..,..),
    EXITS(IEFU83,IEFU84,.....,.....,),.....,.....)
```

Including the exits in SUBSYS(STC) is required and is omitting it is often the cause of the exits not being invoked.

IBM provides default IEFU83/84 modules in SYS1.LPALIB which are merely IEFBR14s, so there should be no problem activating these exits if they have not been used before. A SET SMF=xx command reloads the SMF options from SYS1.PARMLIB(SMFPRMxx).

IEFU83 is required for FTP server records, whereas IEFU84 is required for client records. It is recommended to enable both.

Install SMF Exits

Update SYS1.PARMLIB(PROGxx) to load the exits via the dynamic exit facility e.g.

```
EXIT ADD EXITNAME(SYS.IEFU83) MODNAME(FTPC11?)
EXIT ADD EXITNAME(SYS.IEFU84) MODNAME(FTPC11?)
```

In the above example the '?' should be replaced by an '8' or '9' depending on whether SMF 118 or 119 records are in use. Both FTPC118 and FTPC119 can be installed, if desired.

Activate SMF Exits

The SMF exits can be dynamically activated via the SETPROG operator command e.g.

`SETPROG EXIT,ADD,EXITNAME=SYS.IEFU83,MODNAME=FTPC11?` – '?' is '8' or '9'
`SETPROG EXIT,ADD,EXITNAME=SYS.IEFU84,MODNAME=FTPC11?` – '?' is '8' or '9'

Alternatively an IPL can be instigated to activate the SMF exits.

Verify Functionality

To verify that FTP client records are being processed, the TSO FTP command can be used to initiate an FTP to a remote system. Similarly a transfer to the mainframe from a remote system can be initiated to verify that FTP server SMF records are being processed.

On completion of these transfers, output similar to that in *FTPC118 Output in SMF Exit Mode—JES2 Joblog/SYSLOG* or *FTPT119 Output in TEST Mode—SYSPRINT DD* should be present in the SYSLOG and JES2 joblog of the task (FTP client only).

Chapter 2—Components

The constituent components of the FTPC118/119 programs are described here for reference purposes. Instructions for re-assembling and re-compiling the source code is also included.

Assembler Components

MLWTO

MLWTO is an LE compliant assembler function to issue an operating system Write-To-Operator (WTO) from IBM C. The WTO will either be single-line or multi-line depending on the length of the input.

MLWTO receives a single parameter whose address is passed in R1. The first two bytes of the parameter indicate the length of the WTO text. The remaining bytes of the parameter are the variable length WTO text itself.

MLWTO attempts to determine whether it is running authorized. This is important for two reasons.

1) We have to determine whether we are allowed to issue branch-entry rather than SVC-entry WTO. In certain environments, e.g. some system exits, we are forbidden from issuing SVCs of any kind.
2) If we are 'authorized' then the length of a WTO is increased by 1. This means that single-line WTOs are 127 chars long when authorized. They are 126 chars long when not authorized. MLWTO lines are 71 chars long when authorized and 70 chars when not. Knowing the length of the WTO is important when determining whether to issue an SLWTO or MLWTO and when splitting the lines of an MLWTO.

A single line WTO is issued if the input is 126 (127 auth) characters or less. If the input is longer, then multiple (up to 255) lines of an MLWTO with line length 70 (71 auth) are built. In building the MLWTO, consideration is given to not splitting 'words' of the input over two lines.

The source for MLWTO is provided in `(userid).CBT.FILE600.ASM.PDS(MLWTO)` and a sample job to assemble it is in `(userid).CBT.FILE600.ASM.PDS(ASMOBJ)`. For usage in a C program, the object module would be explicitly included in a link-edit e.g. `(userid).CBT.FILE600.C.PDS(FTPC119L)`.

The program could be adapted for non-LE usage by changing the prologue and epilogue to use MVS-standard, rather than LE-specific linkage.

@@XGET/@@XFREE

@@XGET and @@XFREE are virtual storage allocation subroutines for Systems Programming C (SPC) which use the STORAGE macro. The default (brainless) IBM routines use SVC 120 which would prevent SPC from being used in system exits where SVCs are not allowed.

On entry R0 specifies the length of storage to be obtained, with the high bit on for storage required above the line. On exit R0 indicates the length of storage obtained and R1 its address.

The programs do rudimentary checks to see whether they are running authorized so they can use the preferred high private subpool 230, where possible.

The source for @@XGET/@@XFREE is provided in `(userid).CBT.FILE600.ASM.PDS(EDCXFGS)` and a sample job to assemble them is in `(userid).CBT.FILE600.ASM.PDS(ASMOBJ)`. For usage in a SPC program, the object modules would be explicitly included in a link-edit to override the default routines in `CEE.SCEESPC` e.g. `(userid).CBT.FILE600.C.PDS(FTPC119L)`.

SPRNTLL

SPRNTLL is a function to convert a 64 bit signed integer (*Long Long* type in C) to printable EBCDIC. This is required because Systems Programming C does not support the *Long Long* type in the sprintf() function. IBM raised APAR *PQ42591* for me to document that they have copped out of providing this support.

Two parameters are passed. The first is the address of the output area for the printable 64 bit integer (min length 21 chars). The second is the 64 bit integer itself (*Long Long* type in C). The range of a 64 bit signed integer is -9223372036854775808 to 9223372036854775807. For unsigned integers, only values up to 9223372036854775807 (0x7FFFFFFFFFFFFFFF or 63 bits) will be formatted correctly.

The presence of z/Architecture is checked. If not present a null string is returned. The 64 bit CVDG instruction is used to convert the integer to decimal, then it is split into two portions, unpacked and moved to the output area. Leading zeroes are suppressed.

SPRNTLL is only used in FTPC119 because it has been written for z/Architecture. It cannot be used in FTPC118 because FTPC118 is intended to work on pre-z/Architecture OS/390 releases.

SPRNTLL does not ‘return’ the string value in the conventional C sense. The reason is that I could not work out how to return a string value to C from an assembler function. The documentation in the IBM manuals is unclear and none of my experimentation worked. Consequently using SPRNTLL requires the first parameter of the function to be used in 2 steps in the C code e.g.

```
long long lli1=912475465673LL;
char ch1[21];
sprntll(ch1,lli1);
printf("%s%s","SPRNTLL - integer=",ch1);
```

The source for SPRNTLL is provided in `(userid).CBT.FILE600.ASM.PDS(SPRNTLL)` and a sample job to assemble it is in `(userid).CBT.FILE600.ASM.PDS(ASMOBJ)`. For usage in a C program, the object module would be explicitly included in a link-edit e.g. `(userid).CBT.FILE600.C.PDS(FTPC119L)`.

FTPCTST

FTPCTST is used to test the MLWTO functions of the C SMF exits FTPC118/119. This allows the SMF exit functions of these programs to be tested without installing them as SMF exits.

FTPCTST is passed a single parameter indicating which SMF exit program to call. This will be either FTPC119 (default), FTPC118, FTPSMFEX, or possibly something else.

For each record in the SMFDD input file, the indicated SMF exit program is directly branched to, passing the SMF record which has been read as a parameter. The MLWTO functions of the FTP SMF exits will be invoked and the MLWTOs should appear in the joblog of the job running FTPCTST. The output will be in the same format as if the SMF exits been installed.

FTPCTST can run either APF authorized or not (the program will check). It is better to run it from an APF authorized library as this will test the full length MLWTOs and the authorized STORAGE subpool used in the SMF exits. If run unauthorized the MLWTOs will be one character shorter than they would be in an SMF exit.

The source for FTPCTST is provided in `(userid).CBT.FILE600.ASM.PDS(FTPCTST)` and a sample job to assemble and link it is in `(userid).CBT.FILE600.ASM.PDS(ASMLKED)`. Sample execution JCL is provided in `(userid).CBT.FILE600.ASM.PDS(FTPCTSTJ)`.

C Components

FTPC118

FTPC118 is a C language program which processes SMF type 118 FTP records as described extensively in this document .The program has two modes of operation, selected by compile options EXITMODE and TESTMODE, and a separate link-edit for each mode.

In EXITMODE, FTPC118 uses Systems Programming C facilities to run as an SMF IEFU83/84 exit or a TCPIP FTPSMFEX exit to format and WTO the SMF FTP records.

In TESTMODE, FTPC118 (load module FTPT118) runs as a standard C main program in batch to print the SMF FTP records to standard output.

FTPC118 should work on OS/390 2.5 and above systems where SMF 118 FTP records are enabled.

EXITMODE

In EXITMODE at entry R1 points to the address of the SMF record being written by the system. The __xregs() SPC function is used to determine the value of R1 passed. The chkrec() function is called to verify that it is a valid type 118 FTP record. Type 118 subtypes 3 and 70-75 are processed. All other records are ignored and control returns to the system with RC=0 to allow the SMF record to be written.

For a valid FTP record, an output area is allocated and dartn() is called to format the SMF data to be suitable for WTO. The data fields processed in the FTP Client (C), FTP Server (S) and FTP Server logon failure (L) records are as shown in *Appendix B—Explanation of Output Fields*.

Once formatted, the data is passed to an external assembler subroutine to issue the WTO to the operating system. The format of the output in the syslog will be similar to that in *FTPC118 Output in SMF Exit Mode—JES2* Joblog/SYSLOG.

Once the WTO is issued the program frees storage for the output area and returns to the operating system with RC=0 to allow the SMF record to be written.

TESTMODE

In TESTMODE a 32K buffer is allocated for processing the SMF records. The SMFDD DD is opened as an input file and an SMF record is read from it. The chkrec() function is called to verify that it is a valid type 118 FTP record. Type 118 subtypes 3 and 70-75 are processed. All other records are ignored and control returns to the system. Rejected records are reported to STDERR output.

For a valid FTP record an output area is allocated and dartn() is called to format the SMF data to be suitable for output. The data fields are identical to those described above for EXITMODE.

Once formatted, the data is output to STDOUT via the printf() function. The formatting is almost identical to that described above for EXITMODE. The difference being that the data is prefixed by the system and date/time from the SMF record.

Control then returns back, another SMF record is read and processed and so on until end of file. At EOF the SMFDD is closed, storage is freed and control returns to the operating system.

Compilation

The source is provided in `(userid).CBT.FILE600.C.PDS(FTPC118)` and if the C compiler is available, sample JCL `(userid).CBT.FILE600.C.PDS(FTPC118C)` can be used to compile the program. two object modules are output: FTPC118 for EXITMODE and FTPT118 for TESTMODE.

The FTPC118 compile is for SPC (Systems Programming C) and must specify the NOSTART option. It must not specify RENT, since RENT is not supported in SPC. The code is naturally re-entrant in any case.

The FTPT118 compile is a standard C main program compile and can specify TEST for use with the LE debugger.

On systems with both the OS/390 R10 and z/OS 1.2 compilers, it is possible to select the desired compiler by setting the required JCLLIB statement in the compile JCL `(userid).CBT.FILE600.C.PDS(FTPC118C).`

FTPC119

FTPC119 is a C language program which processes SMF type 119 FTP records as previously described .The program has two modes of operation, selected by compile options EXITMODE and TESTMODE, and a separate link-edit for each mode.

In EXITMODE, FTPC119 uses Systems Programming C facilities to run as an SMF IEFU83/84 exit to format and WTO the SMF FTP records.

In TESTMODE, FTPC119 (load module FTPT119) runs as a standard C main program in batch to print the SMF FTP records to standard output.

FTPC119 should work on z/OS 1.2 and above systems where SMF 119 FTP records are enabled.

EXITMODE

In EXITMODE on entry R1 points to the address of the SMF record being written by the system. The __xregs() SPC function is used to determine the value of R1 passed. The chkrec() function is called to verify that it is a valid type 119 FTP record. Type 119 subtypes 3/70/72 are processed. All other records are ignored and control returns to the system with RC=0 to allow the SMF record to be written.

For a valid FTP record an output area is allocated, dartn() is called then a subtype-specific function is called to format the SMF data to be suitable for WTO. The data fields processed in the FTP Client (C), FTP Server (S) and FTP Server logon failure (L) records are as shown in *Appendix B—Explanation of Output Fields*.

Once formatted, the data is passed to an external assembler subroutine to issue the WTO to the operating system. The format of the output in the syslog will be similar to that in *FTPC119 Output in SMF Exit Mode—JES2 Joblog/SYSLOG*.

Once the WTO is issued the program frees storage for the output area and returns to the operating system with RC=0 to allow the SMF record to be written.

TESTMODE

In TESTMODE the –r and –e switches can be passed as parameters to the program. The –r switch enables TCP/IP domain name resolution of IP addresses in the input data. The –e switch merely echoes the switch options to STDOUT.

A 32K buffer is allocated for processing the SMF records. The SMFDD DD is opened as an input file and an SMF record is read from it. The chkrec() function is called to verify that it is a valid type 119 FTP record. Type 119 subtypes 3/70/72 are processed. All other records are ignored and control returns to the system. Rejected records are reported to STDERR output.

For a valid FTP record an output area is allocated, dartn() is called then a subtype-specific function is called to format the SMF data to be suitable for output. The data fields are identical to those described above for EXITMODE.

Once formatted, the data is output to STDOUT via the printf() function. The formatting is almost identical to that described above for EXITMODE. The differences being that the data is prefixed by the system and date/time from the SMF record, and the IP addresses can optionally (-r switch) include the resolved TCP/IP host name

Control then returns back, another SMF record is read and processed and so on until end of file. At EOF the SMFDD is closed, storage is freed and control returns to the operating system.

Compilation

The source is provided in `(userid).CBT.FILE600.C.PDS(FTPC119)` and if the C compiler is available, sample JCL `(userid).CBT.FILE600.C.PDS(FTPC119C)` can be used to compile the program. two object modules are output: FTPC119 for EXITMODE and FTPT119 for TESTMODE.

The FTPC119 compile is for SPC (Systems Programming C) and must specify the NOSTART option. It must not specify RENT, since RENT is not supported in SPC. The code is naturally re-entrant in any case.

The FTPT119 compile is a standard C main program compile and can specify TEST for use with the LE debugger. The RENT compile option is included to provide constructed reentrancy of the writable static storage used only in TESTMODE.

On systems with both the OS/390 R10 and z/OS 1.2 compilers, it is possible to select the desired compiler by setting the required JCLLIB statement in the compile JCL `(userid).CBT.FILE600.C.PDS(FTPC119C).`

C FTP SMF Record Headers

IBM provides assembler DSECTs for type 118 and 119 records in the `TCPIP.SEZACMAC` members `EZASMF76` and `EZASMF77` respectively. IBM also provides the assembler DSECT–C header mapping utility—CBC3DSCT.

This utility has been used to provide C headers for the type 118 and 119 FTP records in `(userid).CBT.FILE600.C.PDS` as follows.

`DS118R10` — Type 118 record header for OS/390 R10 and above.
`DS118R9` — Type 118 record header for OS/390 R9 (and below?).
`DS119R12` — Type 119 record header for z/OS V1.2 (and above).

The compilation for FTPC118 uses the __LIBREL__ predefined macro name to determine which type 118 header to include for the target operating system.

JCL members `EDCDS118` and `EDCDS119` are provided in `(userid).CBT.FILE600.C.PDS` to reproduce the C headers if needed.

For the type 119 header the output dataset must have sufficient record length for IBM's rather long field names. An FB80 dataset is not sufficient. For this reason, the supplied `(userid).CBT.FILE600.C.PDS` is VB255.

SPRNTLLT

SPRNTLLT is a simple C main program to test/demonstrate the SPRNTLL assembler function. There are no parameters. The program runs and calls SPRNTLL to format the max and min 64 bit signed integers. It then printf()'s the results.

The source is provided in `(userid).CBT.FILE600.C.PDS(SPRNTLLT)` and sample JCL `(userid).CBT.FILE600.C.PDS(SPRNTLLC)` can be used to compile, load and run the program.

WTOT

WTOT is a simple C main program to test/demonstrate the MLWTO assembler function. There are no parameters. The program runs and calls MLWTO to demonstrate both single and multi line WTOs.

The source is provided in `(userid).CBT.FILE600.C.PDS(WTOT)` and sample JCL `(userid).CBT.FILE600.C.PDS(WTOTC)` can be used to compile, load and run the program.

Appendix A—Sample Output

The following are output samples from FTPC118/119 but not all possible subtype/mode combinations are shown.

FTPT118 Output in TEST Mode—SYSPRINT DD

SMF record type 118 subtype 3 (FTP client)

```
TEST 2002/10/02 07:52:04.92 FTPC118-02I FTP Client cm=STOR lr=226 cp=1096 cf=21 sa=10.1.160.220
sl=10.13.80.11 su=troon ASCII Stream File Seq trs=07:52:04.65 dur=00:00:00.02 tbc=5924 rt=296.20kb/sec
hst=tso03 stc=AWCPAW dsn=AWCPAW.TEMP.TXT
```

SMF record type 118 subtype 71 (FTP server delete)

```
B303 2002/07/29 04:58:31.69 FTPC118-01I FTP Server cm=DELE ty=SEQ lr=250 rp=2892 lp=21 sa=10.10.8.36
sl=10.1.160.194 su=ARKO02 trs=04:58:31.69 hst=tso01 dsn=ARK.CABIN.CBA0729
```

SMF record type 118 subtype 72 (FTP server logon failure)

```
TEST 2002/10/02 10:08:28.34 FTPC118-01I FTP Server cm=LOGN lr=530 rp=3238 lp=21 sa=10.13.80.11
sl=10.1.160.225 su=AWCPAW trs=10:08:28.33 hst=tso03
```

SMF record type 118 subtype 73 (FTP server rename)

```
B303 2002/07/29 00:33:22.46 FTPC118-01I FTP Server cm=REN ty=SEQ lr=250 rp=2529 lp=21 sa=10.10.8.55
sl=10.1.160.190 su=ARKT01 trs=00:33:22.46 hst=tso01 dsn=BPJ.ATBS.SIMS.G0001V00
ds2=ZACXL.USERDATA.SIMS.G0004V00
```

SMF record type 118 subtype 74 (FTP server retrieve)

```
B303 2002/07/29 05:05:10.89 FTPC118-01I FTP Server cm=RETR ty=SEQ lr=250 rp=1040 lp=21 869
sa=10.13.15.231 sl=10.1.160.223 su=AWCPAW ASCII Stream File Seq trs=20:38:44.44 dur=00:00:00.04
tbc=1762 rt=44.05kb/sec hst=tso03 dsn=AWCPAW.TEMP.TXT
```

SMF record type 118 subtype 75 (FTP server store)

```
TEST 2002/10/02 10:08:45.93 FTPC118-01I FTP Server cm=STOR ty=SEQ lr=250 rp=3238 lp=21 sa=10.13.80.11
sl=10.1.160.225 su=AWCPAW ASCII Stream File Seq trs=10:08:45.87 dur=00:00:00.05 tbc=5924
rt=118.48kb/sec hst=tso03 dsn=AWCPAW.TEMP.TXT
```

FTPC118 Output in SMF Exit Mode—JES2 Joblog/SYSLOG

SMF record type 118 subtype 3 (FTP client) – JES2 Joblog output

```
08.29.22 JOB62645  FTPC118-02I FTP Client cm=RETR lr=226 cp=1064 cf=21 sa=10.1.160.220  335
   335             sl=10.1.43.57 su=anonymou ASCII Stream File Seq trs=08:29:18.11
   335             dur=00:00:04.23 tbc=174988 rt=41.37kb/sec hst=tso03 stc=AWCPAW
   335             dsn=SYSM.OS390.HOLDDATA(MONTH)
```

SMF record type 118 subtype 75 (FTP server store) – SYSLOG output

```
14:17:42.81 STC62655 00000000 FTPC118-01I FTP Server cm=STOR ty=SEQ lr=250 rp=3242 lp=21 843
                843 00000000 sa=10.13.80.11 sl=10.1.160.220 su=AWCPAW ASCII Stream File Seq
                843 00000000 trs=14:17:42.66 dur=00:00:00.05 tbc=1764 rt=35.28kb/sec hst=tso03
                843 00000000 dsn=AWCPAW.TEMP.TXT
```

FTPT119 Output in TEST Mode—SYSPRINT DD

SMF record type 119 subtype 3 (FTP client)

```
TEST 2002/07/31 07:09:22.61 FTPC119-02I FTP Client cm=RETR lr=226 cp=1082 cf=21 sa=10.1.160.221
sl=10.1.43.57 su=anonymou ASCII Stream File Seq trs=07:09:12.43 dur=00:00:10.07 tbc=177448
rt=17.62kb/sec hst=tso03 stc=AWCPAW asn=AWCHOLDD dsn=SYSM.OS390.HOLDDATA(MONTH)
```

SMF record type 119 subtype 70 (FTP server transfer completion)

```
TEST 2002/08/31 08:02:44.25 FTPC119-01I FTP Server cm=RETR ty=JES lr=125 rp=3124 lp=21 sa=10.13.80.148
sl=10.1.160.223 su=AWCPAW ASCII Stream File Seq trs=08:02:44.24 dur=00:00:00.01 tbc=1792
rt=179.20kb/sec hst=tso03.dha.aramco.com.sa dsn=AWCPAW.AWCPAWS.JOB64311.D0000003.JESJCL
```

SMF record type 119 subtype 72 (FTP server logon failure)

```
TEST 2002/09/01 07:33:39.29 FTPC119-03I FTP Server cm=LOGN lp=21 rp=3797 sa=10.1.160.225
sl=10.13.80.182 su=AWCPAW reas=1(password not valid)
```

FTPT119 Output in TEST Mode with -r switch—SYSPRINT DD

SMF record type 119 subtype 3 (FTP client)

```
TEST 2002/07/31 07:09:22.61 FTPC119-02I FTP Client cm=RETR lr=226 cp=1082 cf=21
sa=tso03.dha.aramco.com.sa(10.1.160.221) sl=cisproxy.dha.aramco.com.sa(10.1.43.57) su=anonymou ASCII
Stream File Seq trs=07:09:12.43 dur=00:00:10.07 tbc=177448 rt=17.62kb/sec hst=tso03 stc=AWCPAW
asn=AWCHOLDD dsn=SYSM.OS390.HOLDDATA(MONTH)
```

FTPC119 Output in SMF Exit Mode—JES2 Joblog/SYSLOG

SMF record type 119 subtype 3 (FTP client) – JES2 Joblog output

```
08.29.22 JOB62645  FTPC119-02I FTP Client cm=RETR lr=226 cp=1064 cf=21 sa=10.1.160.220  334
   334             sl=10.1.43.57 su=anonymou ASCII Stream File Seq trs=08:29:18.11
   334             dur=00:00:04.23 tbc=174988 rt=41.37kb/sec hst=tso03 stc=AWCPAW
   334             asn=AWCHOLDD dsn=SYSM.OS390.HOLDDATA(MONTH)
```

SMF record type 119 subtype 70 (FTP server transfer completion) – SYSLOG output

```
14:17:42.79 STC62655 00000000 FTPC119-01I FTP Server cm=STOR ty=SEQ lr=250 rp=3242 lp=21 842
                842 00000000 sa=10.13.80.11 sl=10.1.160.220 su=AWCPAW ASCII Stream File Seq
                842 00000000 trs=14:17:42.66 dur=00:00:00.05 tbc=1764 rt=35.28kb/sec
                842 00000000 hst=tso03.dha.aramco.com.sa dsn=AWCPAW.TEMP.TXT
```

Appendix B—Explanation of Output Fields

The field names are derived from the last 2 or 3 characters of the field name in the SMF type 118 DSECT. For consistency these names have for the most part been retained in FTPC119 despite the SMF type 119 DSECT field names being entirely different.

Report Field	Client, Server, Logon	Description	Type 118 Name	Type 119 Name	Example	Notes
cm=	C/S/L	Subcommand (RFC959)	SMFFTPCM	C: smf119ft_fccmd S: smf119ft_fscmd L: (derived)	cm=STOR	
ty=	S	File type	SMFFTPTY	smf119ft_fsftype	ty=SEQ	
lr=	C/S/L	Server last reply to client	C: SMFFTPTY S: SMFFTSLR L: SMFFTSLR	C: smf119ft_fclreply S: smf119ft_fslreply L: (not present)	lr=250	1
cp=	C	Local port	SMFFTPCP	smf119ft_fcclport	cp=1064	
cf=	C	Foreign port	SMFFTPCF	smf119ft_fccrport	cf=21	
rp=	S/L	Remote port	SMFFTSRP	S: smf119ft_fscrport L: smf119ft_ffrport	rp=4608	
lp=	S/L	Local port	SMFFTSLP	S: smf119ft_fsclport L: smf119ft_fflport	lp=21	
sa=	C/S/L	C: Local IP address S: Remote IP address L: Remote IP address	SMFFTPSA	C: smf119ft_fcclip S: smf119ft_fsclip L: smf119ft_ffrip	sa=10.13.80.11 or sa=bc331172.aramco.com(10.13.80.11)	9,10
sl=	C/S/L	C: Remote IP address S: Local IP address L: Local IP address	SMFFTPSL	C: smf119ft_fccrip S: smf119ft_fscrip L: smf119ft_fflip	sl=10.1.160.220 or sl=tso03.dha.aramco.com.sa(10.1.160.220)	9,10
su=	C/S/L	C: Remote userid S: Local userid L: Local userid	SMFFTPSU	C: smf119ft_fcruser S: smf119ft_fssuser L: smf119ft_ffuserid	su=AWCPAW	
	C/S	Data format	SMFFTPFM	C: smf119ft_fctype S: smf119ft_fstype	ASCII	3
	C/S	Mode	SMFFTPMO	C: smf119ft_fcmode S: smf119ft_fsmode	Stream	3
	C/S	Structure	SMFFTPST	C: smf119ft_fcstruct S: smf119ft_fsstruct	File	3
	C/S	Dataset type	SMFFTPDT	C: smf119ft_fcdstype S: smf119ft_fsdsttype	Seq	3
trs=	C/S/L	Transmission start time	SMFFTTRS	C: smf119ft_fcstime S: smf119ft_fsstime L: (not present)	trs=08:09:08.87	1
dur=	C/S	Transmission duration	(derived)	C: smf119ft_fcdur S: smf119ft_fsdur	dur=00:00:00.00 dur=n/a	3,5,11
tbc=	C/S	Transmission byte count	SMFFTTBC SMFFTBYF	C: smf119ft_fcbytes S: smf119ft_fsbytes	tbc=5924	3,6,7
rt=	C/S	Transmission rate	(derived)	(derived)	rt=296.20kb/sec rt=n/a	3,12
hst=	C/S/L	Host name	SMFFTHST	C: smf119ft_fchostname S: smf119ft_fshostname L: (not present)	hst=tso03	1,4
stc=	C	Local userid	SMFFTSTC	smf119ft_fcluser	stc=AWCPAW	
asn=	C	Address space name	(not present)	smf119ti_asname	asn=AWCPAW	2
dsn=	C/S	1st Dataset name	SMFFTDSN SMFFTMEM SMFFTVAR	C: smf119ft_fcfilename S: smf119ft_fsfilename1	dsn=AWCPAW.TEMP.TXT	

(continued)

Report Field	Client, Server, Logon	Description	Type 118 Name	Type 119 Name	Example	Notes
ds2=	S	2nd Dataset name	SMFFTDS2 SMFFTMM2 SMFFTVAR	smf119ft_fsfilename2	ds2=AWCPAW.TEMP.TXT	8
reas=	L	Logon failure reason	(not present)	smf119ft_ffreason	reas=1(password not valid)	2

Notes:
1) For Logon (L type) records the field is only present in type 118 records
2) Type 119 records only
3) For Server (S type) records the field is not present for renames and deletes
4) For type 119 Server (S type) records the host name includes the domain name
5) For type 118 records the duration is derived and has a maximum value of 23:59:59.99
6) On pre-OS/390R10 systems the byte count has a maximum value of 4294967295
7) For type 118 records on OS/390R10 or above systems, byte counts > 429497295 are expressed as a gigabyte (Gb) value
8) Renames only
9) For type 119 records IPv6 format addresses are supported
10) Domain name only included for type 119 records with IPv4 addresses and when the program is run in TESTMODE (FTPT119) with the –r switch included
11) For type 119 server records n/a will be indicated for duration when the transfer spans midnight and the fix for APAR PQxxxxx is not applied
12) n/a will be indicated for rate when the duration is zero or n/a

Appendix C—FTPSMFEX

IBM TCPIP provides a standard exit point for processing the FTP SMF records before they are written to SMF, which is named FTPSMFEX. The data passed to FTPSMFEX is almost identical to that available in the operating system SMF exits for type 118/119 records. FTPC118 provides an alternate entry point for FTPSMFEX.

FTPSMFEX can be used by those wishing to log FTP traffic but who do not wish to install the operating system SMF exit FTPC118. However, it is recommended to use FTPC118 and not FTPSMFEX for the following reasons.

- FTPSMFEX only receives control for server SMF records and not for client records.
- FTPSMFEX is only available for type 118 records and not for type 119 records. Installations should use type 119 records, if available, since type 118 records are not likely to be enhanced in future by IBM.
- There can only be one FTPSMFEX in the system, whereas the operating system dynamic exit facility allows multiple IEFU83/84 exits.
- In the *z/OS Communications Server IP Configuration Reference* manual it states that SPC is not supported for FTP server exits, although it does not indicate whether this includes SPC applications that operate without the LE library.
- There is a possible conflict with the IBM product *Tivoli NetView Performance Monitor for TCP/IP (5698-PMI)* which uses FTPSMFEX.
- The FTPSMFEX contained herein has not been as extensively tested as FTPC118.

For the above reasons, some installations may want to exclude the FTPSMFEX alias of FTPC118 when copying the load modules in *Copy SMF Exit Programs to a Linklist Library*.

Activating FTPSMFEX

To activate FTPSMFEX, make the load module available via an APF authorized linklist library and specify it in `SYS1.TCPPARMS(FTPSDATA)` or local equivalent.

- `SMFEXIT; SMF exit FTPSMFEX`

The FTP server should be recycled to change this option.

Appendix D—Systems Programming C for System Exits

> *"C combines the power of assembly language with the flexibility of assembly language."*
>
> ***The unknown programmer***

Overview of SPC

C is an excellent choice for writing operating system code on most platforms because of the power of the language and its widespread acceptance. On MVS however, customer-written additions to the operating system are generally written in assembler language. Many such routines are in use and it is still a requirement at most sites for the Systems Programmers to know assembler in order to maintain them. Even so, it may be little-known fact that with Systems Programming C (SPC) it is possible to write MVS operating system exits in C.

There are several flavors of SPC. For example, applications can be created with persistent C environments for improved performance if an application is continually re-invoked, and it is possible to create free-standing applications which operate without LE. This document is concerned with creating system exits in SPC as a substitute for assembler code.

Recommendations

The following are recommendations for using SPC in system exits and are based on my own experiences in writing FTPC118/119. However, circumstances differ and these recommendations may not be valid in all situations.

- **Specify `#pragma environment(xxxxxxxx,nolib)`**— Many exits have fixed load module and/or entry point names, so '`xxxxxxxx`' should specify the entry point name. This statement also enables alternate entry points to be created. Almost certainly it will not be possible to use the LE library in a system exit application, since LE uses operating system services such as SVCs, which are not allowed in many system exits. The '`nolib`' keyword should be specified to ensure that the LE library is not used.
- **Write naturally re-entrant code**—System exits are usually required to be re-entrant so that they can be loaded in LPA and potentially executed concurrently by multiple tasks. The '`RENT`' compiler option is not available in SPC, so program code must be naturally re-entrant. This precludes the use of writable static program variables i.e. those with the static or external storage class and writable strings.
- Replace **`@@XGET/@@XFREE`**—These routines are called whenever virtual storage is allocated or freed. Unfortunately the IBM versions of these routines use SVC120, and in a system exit an abend0F8 usually results. You need to supply your own routines which do not use SVCs.
- **Specify `#pragma runopts("TRAP(OFF)")`**—Standard LE ESTAE and ESPIE is not recommended in a system exit so LE recovery should be turned off.

- **Specify `#pragma runopts("HEAP(xK,xK,ANY),STACK(xK,xK,ANY)")`**—The defaults for HEAP and STACK cannot be relied upon as documented in APAR *PQ47946*, so control them directly yourself .
- **Use only available functions**—Operating without the LE library restricts the functions available in the program to those which are either built-in to the compiler, or a handful of library functions provided with SPC. These are summarized as follows.

Built-in functions	Mathematical—*abs(), fabs()* Memory manipulation—*memchr(), memcmp(), memcpy(), memset(), cds(), cs()* String operations—*strcat(), strchr(), strcmp(), strcpy(), strlen(), strrchr()*
Memory management functions	*malloc(), calloc(), realloc(), free()*
Other functions	*exit(), sprintf()*
SPC functions	*__xregs(), __xusr(), __xusr2(), __24malc(), __4kmalc()*

- **Provide your own C library functions**—Many of the standard C library functions are unavailable without the LE library. However, they sometimes can be substituted by C code from other sources. For example, I obtained a source version of div() from NETBSD.
- **Be prepared to write some assembler**—If any operating system services are needed, it is likely that some sort of assembler wrapper routine will be required. In addition, substituting for unavailable C library functions may involve writing assembler code. These limitations may defeat the object of writing the exit routine in SPC.
- **Ensure SPC is supported for target application**—There are cases where SPC is specifically not supported for a particular exit. This may be in situations within some products where some sort of LE environment is active or where some flavor of SPC is already used. For example it states in the *z/OS Communications Server IP Configuration Reference* that SPC is not supported for the FTP server exits. However, no reason is given, and it does not state whether this restriction also includes SPC applications operating without the LE library.
- **`sprintf()` may not be all it seems**— Some format modifiers are not available in SPC—see APARs *PQ42591* and *PQ45794*. Also, ensure that EDCXSPRT (SPC version of sprintf) is included at link-edit.
- **Specify `#include <spc.h>`**—This contains prototypes of SPC functions.
- **Use `CEE.SCEESPC`**—This link-edit SYSLIB dataset contains the SPC-specific functions and the SPC versions of some library functions. This dataset should the first and only LE library in the link-edit SYSLIB to ensure that the LE library is not inadvertently included.

IBM APARs Raised

During the development of FTPC118/119 I encountered a number of problems related to SPC for which APARs were raised, only one of which IBM were willing to develop a fix for, and then only a partial fix. Usually I had to code around the problem or simply avoid using the feature I was trying to use. IBM seemed reluctant to develop fixes for SPC and I guess that may be because not many of their customers are using it.

One APAR was also raised in March 2004 for TCP/IP.

PQ42591

This was a problem relating to an unresolved symbol during link-edit, when including EDCXSPRT (SPC sprintf). It had been caused by the addition of 64 bit integer support to sprintf and the problem was encountered when we upgraded to OS/390 R9. The following message in the link edit SYSPRINT output details the problem.

```
IEW2456E 9207 SYMBOL @@DIVU64 UNRESOLVED.
```

IBM provided a fix for the link-edit problem but documented in the *OS/390 C/C++ Programming Guide* manual that they would not provide support for 64 bit integers in SPC sprintf. This was most unfortunate because OS/390 R10 provided a 64 bit field for bytes transferred in FTP SMF records. I wanted to use this field, but because of this APAR, I was forced to write my own sprintf equivalent routine for 64 bit integers—called *SPRNTLL*.

Update June 2004 – After installing z/OS 1.4 the problem came back. We raised a PMR with IBM expecting a z/OS 1.4 version of this APAR to be generated. However, we were bitterly disappointed because IBM refused to recognize that there was a problem. They stated that the simple workaround was to add the CEE.SCEELKED library to the SYSLIB of the link-edit. I was not happy with this because routines from CEE.SCEELKED are not safe for inclusion in a SPC load module intended to run in a system exit. IBM were adamant that they weren;t going to fix it, so I reluctantly change the link edit jobs FTPC118L and FTPC119L to add the CEE.SCEELKED library.

PQ45794

In OS/390 R10 IBM also provided a floating point field for bytes transferred in FTP SMF records. Before I wrote *SPRNTLL*, I experimented with floating point (FP) formatting using this field with sprintf. I thought that surely if I couldn't use 64 bit integer formatting with sprintf under SPC then I **could** use FP formatting.

But unfortunately IBM didn't agree. In APAR PQ45794 they decided that FP formatting with SPC sprintf was not supported when not using the LE library. Since I can't use the LE library in system exits, I was stymied again!

Some code in FTPC118 still uses FP formatting with sprintf but the conditional compilation ensures that this is only done in TEST mode. I decided not to incorporate *SPRNTLL* into FTPC118 since it was written for z/Architecture only.

PQ47946

Conscientiously I decided to check the virtual storage used by FPTC118/119 in SMF exit mode. Since they are written in C, I do not necessarily have full control over the amount of storage allocated and as SMF exits they are running in multiple address spaces. I therefore wanted to verify that they were not using excessive storage.

I was right to be careful! Using a storage GTF trace I found that storage allocations for 64Kb and 512Kb were being issued. I knew this was much more than was really needed.

In the *OS/390 C/C++ Programming Guide* manual it states that the initial STACK size is 'the minimum size required to start the C program' and the initial HEAP size is 12K. Clearly this did not match my findings from the GTF trace, so APAR PQ47946 was raised.

But IBM copped out again. They said that the problem was only related to the TRAP(OFF) run-time option and that I could control the HEAP and STACK sizes with a `#pragma runopts` anyway. Reluctantly I agreed to a FIN closure of the APAR and to code HEAP and STACK sizes in my programs.

As of September 2002 I enquired as to whether this APAR had yet been incorporated into a development release, but IBM said that it hadn't.

PQ87028

Prior to the fix for APAR PQ87028 the *smf119ft_fsdur* (server transfer duration) and *smf119ft_fcdur* (client transfer duration) are invalid if the transfer spans across midnight. The FTPC119 program was changed in March 2004 to show the duration as 'n/a' if an invalid duration is detected.

Appendix E—Other IBM APARs and Product Enhancements

IBM APARs

PQ80090

APAR PQ80090 changed the *smf119ft_fclreply* (last reply to client) field in the SMF type 119 record from a binary value to EBCDIC. The FTPC119 program was changed in February 2004 to allow for both data forms.

PK01634

APAR PK01634 added blank filling for a number of fields in the SMF type 119 record including *smf119ft_fcm1, smf119ft_fsm1* and *smf119ft_fsm2* (member names). A minor amendment to the FTPC119 program was made in June 2005 to accommodate this.

Enhancements

FTPLOGGING in FTP.DATA

In z/OS 1.4 IBM introduced the FTPLOGGING and ANONYMOUSFTPLOGGING statements in the FTP server FTP.DATA configuration file. These enable logging of FTP server activity via the UNIX System Services syslogd daemon. The messages produced provide additional information on the transfer including dataset allocation details and resolved host name. However, they do not detail the transfer duration and rate. The messages can be routed to the conventional JES SYSLOG using a /dev/console specification in the syslogd configuration file.