Customization
Customization

This is a major revision of, and obsoletes, SH12-5482-05.

This edition applies to

- Release 1 Modification Level 1 of NetView File Transfer Program for VSE (5686-013)
- Release 1 Modification Level 1 of NetView File Transfer Program for VM (5684-048)
- Release 2 Modification Level 1 of NetView File Transfer Program for MVS Version 2 (5685-108)

and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers’ comments may be provided at the back of this publication, or you may address your comments to the following address:

IBM Deutschland Entwicklung GmbH
Information Development, Dept. 0446
Postfach 1380
71003 Boeblingen
Germany

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1988, 1994. All rights reserved.

Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.
Chapter 13. Accessing the APL

Chapter 14. Using the APX

Chapter 15. Accessing the APX

Chapter 16. Using REXX Application Programs and User-Exit Routines

REXX Naming Conventions

Defining the Control Blocks

Using the External Functions

Calling NetView FTP V2 MVS from a REXX Application Program

Calling the External Function DVGCXRIA

Invoking a User-Exit Routine from NetView FTP V2 MVS

What DVGCXRIX Does

How to Link the User-Exit Routine

Positioning of Values in the Message Area

External Function DVGCXRMF (REXX)

Special Considerations when Processing the APX Control Block
Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM’s product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of IBM’s intellectual property rights may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user’s responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood NY 10594, U.S.A.

Programming Interfaces

This book is intended to help the customer write user-exit routines for NetView File Transfer Program (abbreviated to NetView FTP). This book primarily documents Product-Sensitive Programming Interface and Associated Guidance Information provided by NetView FTP V2 MVS, NetView FTP VM, and NetView FTP VSE.

Product-Sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of this IBM software product. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-Sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

However, this book also documents General-Use Programming Interface and Associated Guidance Information.

General-Use programming interfaces allow the customer to write programs that request or receive the services of NetView FTP V2 MVS, NetView FTP VM, or NetView FTP VSE.

General-Use Programming Interface and Associated Guidance Information is identified where it occurs, by an introductory statement to a chapter or section.
Trademarks and Service Marks

The following terms, denoted by an asterisk (*) in this publication, are trademarks of the IBM Corporation in the United States or other countries or both:

- AIX
- AIX/6000
- MVS/ESA
- MVS/XA
- NetView
- OS/400
- RACF
- RMF

X  NetView FTP Customization
About This Book

This book describes the user exits of NetView® File Transfer Program (program numbers 5686-013, 5684-048, and 5685-108). It contains information that enables you to customize the NetView File Transfer Program.

Unless indicated otherwise, all text in this book applies to the following:
- NetView FTP V2 MVS
- NetView FTP VSE
- NetView FTP VM.

Abbreviations and Special Terms

Throughout this manual, the following abbreviations are used:

**NetView FTP**
NetView File Transfer Program (regardless of version number and operating system).

**NetView FTP V2 MVS**
NetView File Transfer Program Version 2 for MVS.

**NetView FTP V2.2 MVS**
NetView File Transfer Program Version 2 Release 2 Modification Level 1 for MVS.

**NetView FTP VSE**
NetView File Transfer Program for VSE.

**NetView FTP VM**
NetView File Transfer Program for VM.

**NetView FTP/400**
NetView File Transfer Program for OS/400®.

**NetView FTP AIX®**
NetView File Transfer Program Server for AIX and NetView File Transfer Program Client for AIX.

**NetView FTP/2**
NetView File Transfer Program Server/2 and NetView File Transfer Program Client/2.
When it is not necessary to specify which NetView FTP level is referred to, the term **transfer program** is used.

In addition, unless specifically stated otherwise, this book assumes that both the local system and the remote system have the same program level installed. If the remote system has a different program level installed, the transfer program on the local system can only use the facilities that are compatible with the transfer program on the remote system. See the *Installation, Operation, and Administration* book for your transfer program for details of how your transfer program transfers files to and from a system with a different transfer program.

The term **LU 0 conversation** identifies the type of conversation in transfers between:

- NetView FTP V1 VM and NetView FTP V1 MVS
- NetView FTP V1 VM and FTP V2
- NetView FTP V1 VSE and NetView FTP V1 MVS
- NetView FTP V1 VSE and FTP V2.

This book sometimes refers to a location with an MVS operating system as an “MVS location,” a system with a VSE operating system as a “VSE location,” a system with a VM operating system as a “VM location,” a system with an AIX/6000* operating system as an “AIX workstation”.

In this book, the following groups of terms are used:

- **data set**, **file**, and **cluster** are all used as synonyms.
- **file handler user-exit routine** and **user-written file handler** are used as synonyms.
- **hex** is used as an abbreviation for hexadecimal.

The back of this book provides a glossary, bibliography and index.
Chapter 1. Customizing NetView FTP

NetView FTP fulfills the general requirements of a file transfer communication system without the need for additional coding. However, you can customize NetView FTP, to enhance or extend it to meet the individual requirements of your installation.

You can customize NetView FTP using the following user-written exit routines:

**User-Written File Handler**
To provide support for file types that NetView FTP does not directly support.

**Pre-Queuing User-Exit Routine**
To perform additional checks and to modify special request parameters.

**Pre-Transfer User-Exit Routine**
To perform additional security-checking and special transfer-preparation actions before the file transfer begins.

**Post-Transfer User-Exit Routine**
To perform statistics-collection and special post-processing actions after the file transfer has finished.

**Post-Conversation User-Exit Routine**
To modify the status of a request after the file transfer is completed.

You write a user-exit routine in PL/I, Assembler, REXX, or C Language. When writing a user-exit routine, you must ensure that the linkage conventions are followed. Any pre-transfer, post-transfer, or post-conversation user-exit routine must be reentrant, as a NetView FTP server can process more than one file-transfer request at a time. NetView FTP provides sample user-exit routines and interface routines in the sample library (installation library for VSE). The interface routine allows user-exit routines to be invoked. It must be linked with all user-exit routines. For example, if you want to use a PL/I application program, the entry name in the link control statements must refer to the interface routine (DVGCCPLI for NetView FTP V2 MVS and NetView FTP VM) entry point.

The load module name of a pre-queuing user-exit routine must be DVGCIRQX for NetView FTP V2 MVS and DVGIFRQX for NetView FTP VM or NetView FTP VSE. The load module name of a pre-transfer, post-transfer, or post-conversation user-exit routine can be any valid name except FTPSECUR. The chosen name must be specified as the value of the PPEXIT, PRETRAN, POSTTRAN, or POSTCONV server initialization parameter. See the *Installation, Operation, and Administration* book for your transfer program for details about this parameter.

---

1 PL/I does not apply to NetView FTP VSE.
2 REXX and C language apply to NetView FTP V2 MVS only.
3 VSE uses the term phase name instead of load module name.
4 PRETRAN, POSTTRAN, and POSTCONV apply to NetView FTP V2 MVS only.
This book gives detailed information on how to implement user-exit routines. Before you start to write an exit routine, familiarize yourself with the concepts of NetView FTP by reading the User's Guide and the Installation, Operation, and Administration book for your transfer program.

Special Considerations for MVS/XA or MVS/ESA

User-Written File Handler

When running under MVS/XA* or MVS/ESA*, NetView FTP sets the addressing mode when it passes control to a user-written file handler. The called routine is given control in the addressing mode that the linkage editor indicates in its partitioned data set (PDS) directory entry. Therefore, you can code the file handler for a 24-bit or a 31-bit addressing mode.

On entry to the file handler, the high order bit, bit 0 of Register 14, is set to show the addressing mode of the calling NetView FTP module. If bit 0 is set to 0, the calling module executes in 24-bit addressing mode. If bit 0 is set to 1, the calling module executes in 31-bit addressing mode. All the areas that NetView FTP allocates for use with a user-written file handler reside below the 16MB boundary.

Pre-Queuing User-Exit Routine

When running under MVS/XA, NetView FTP sets the addressing mode when it passes control to a user-exit routine. The called routine is given control in the addressing mode that the linkage editor indicates in its PDS directory entry. Therefore, you can code the user-exit routine for a 24-bit or a 31-bit addressing mode.

Pre-Transfer User-Exit Routine

When running under MVS/XA or MVS/ESA, NetView FTP sets the addressing mode when it passes control to a user-exit routine. The called routine is given control in the addressing mode that the linkage editor indicates in its PDS directory entry. Therefore, you can code the user-exit routine for a 24-bit or a 31-bit addressing mode.

Post-Transfer User-Exit Routine

When running under MVS/XA, NetView FTP sets the addressing mode when it passes control to a user-exit routine. The called routine is given control in the addressing mode that the linkage editor indicates in its PDS directory entry. Therefore, you can code the user-exit routine for a 24-bit or a 31-bit addressing mode.

Post-Conversation User-Exit Routine

When running under MVS/XA, NetView FTP sets the addressing mode when it passes control to a user-exit routine. The called routine is given control in the addressing mode that the linkage editor indicates in its PDS directory entry. Therefore, you can code the user-exit routine for a 24-bit or a 31-bit addressing mode.

5 MB equals 1 048 576 bytes.
Chapter 2. Writing a User-Written File Handler

You write a file-handler user-exit routine to perform I/O operations on types of files that NetView FTP cannot, or does not, gain access to in the way you require. You can incorporate NetView FTP features, such as checkpointing, data compression, and data encryption.6

The file types that your transfer program supports are described in the User’s Guide for your transfer program. When accessing supported file types, a transfer program uses its own internal file handlers.

If the transfer program at either the requesting or responding system cannot handle a particular file type, user-written file handlers must be used at both systems. You cannot use a user-written file handler at one system and a transfer program’s internal file handler at the other system. A file transfer that started using a transfer program’s internal file handler cannot be restarted using a user-written file handler and vice versa.

NetView FTP transfers files on a record basis. NetView FTP’s definition of a record might be different to the definition of a record in the access method used in a user-written file handler.

In NetView FTP, a record is a string of data passed by a file handler to a server. NetView FTP does not try to determine whether the string of data supplied by the file handler is, in the sense of the access method used, a record, a block of records, or only part of a record.

In this chapter, the term “record” means a record as in NetView FTP.

Invoking a User-Written File Handler

You must compile (or assemble) and link-edit a user-written file handler independently of the transfer program. A server loads a file handler into its storage area when it processes a file-transfer request. The server loads a user-written file handler only if the file type is USER, and if the load module name of the file handler is specified in the request. When the server finishes a file transfer, either successfully or unsuccessfully, it deletes the file handler from its storage area.

6 Data encryption applies to NetView FTP V2 MVS only.
A server always uses the same entry point when it calls a file handler. This entry point is passed to a server by a macroinstruction. If the user-written file handler is link-edited with the reusable attribute, this macroinstruction ensures that the transfer program loads only one copy of the user-written file handler. If the user-written file handler is link-edited with the nonreusable attribute, this macroinstruction ensures that the transfer program always loads a new copy of the user-written file handler. This guarantees that you can use any kind of file handler again in the transfer program’s parallel-transfer environment.

The user-written file handler must be link-edited with the reenterable attribute set if more than one server uses the same file handler.

Notes:
1. For MVS and VM: Do not code a user-written file handler so that it is serially reusable and then use it for parallel transfers on the same server. A server passes control to a user-written file handler by a direct branch. This cannot protect a user-written file handler.
2. For MVS: Because a server must be linked to an APF authorized library, you must also link your user-written file handler to an APF authorized library.

Coding a User-Written File Handler

The transfer program calls the specified or default user-written file handler several times during a file transfer to perform various functions (OPEN, GET, PUT, CLOSE, and POINT). For each call, the transfer program passes the name of the requested function to the file handler. Only one function is requested per call. Therefore, write the main skeleton of your user-written file handler as a select-group, where the select argument is the requested function.

A transfer program requests the following functions in a fixed sequence that depends upon the special conditions of the file transfer:

<table>
<thead>
<tr>
<th>Function</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN</td>
<td>Prepare the file for transfer.</td>
</tr>
<tr>
<td>GET</td>
<td>Get a record from the file and pass it to the sending transfer program.</td>
</tr>
<tr>
<td>PUT</td>
<td>Receive a record from the receiving transfer program and write it to the file.</td>
</tr>
<tr>
<td>POINT</td>
<td>Position the file for subsequent processing.</td>
</tr>
<tr>
<td>CLOSE</td>
<td>Complete the file processing.</td>
</tr>
</tbody>
</table>

For the sending system, a user-written file handler must be able to perform the OPEN, CLOSE, and GET functions.
For the receiving system, a user-written file handler must be able to perform the OPEN, CLOSE, and PUT functions.

In addition, if the user-written file handler is required to perform a restart from checkpoint, it must be able to perform the POINT function for both the sending and the receiving system. “Sequence of Function Calls for a User-Written File Handler” on page 34 gives an overview of the function sequences for most common transfer situations.

When you design a user-written file handler, you decide whether it runs in the sending system or in the receiving system. You can write a user-written file handler that runs in both the sending and the receiving systems. In both cases, if there is a user-written file handler running in one system of a file transfer, there must be a compatible user-written file handler running in the other system.

The CALL argument list and the user file-handler parameter list are the main components of the interface between the transfer program and a user-written file handler. The CALL argument list is described in “Using the CALL Argument List” on page 31. The parameter list is described in “Using the User File-Handler Parameter List” on page 23.

Each time the transfer program makes a function call, the FUEINPTR field of the user file-handler parameter list and the second word in the CALL argument list point to a set of parameters. These parameters are specified in the file-transfer request, and contain information that the originator specifically wants the user-written file handler to use. The User’s Guide for your transfer program contains a description of how these parameters are specified.

The format of the parameters passed to a user-written file handler is described in “Presenting Information Parameters to the User-Written File Handler” on page 32.

Note: A user-written file handler must not write into the storage area pointed to by FUEINPTR. Instead, it must copy the parameters into a working buffer, and modify the copied values.

**Considerations when Implementing a User-Written File Handler**

When writing a user-written file handler, keep the following in mind:

- The POINT function is needed only when you plan to implement restart from checkpoint.

  If you use the POINT function you must provide additional specifications in the OPEN and PUT functions, as described in “Designing a User-Written File Handler with Checkpointing” on page 14.

- Take care when implementing an abnormal-termination recovery routine as a function in your user-written file handler. The transfer program has an ESTAE recovery routine (MVS), an ABEND exit routine (VSE), or an ABNEXIT exit routine (VM) that is active when the transfer program calls your user-written file handler.

  Your user-written file handler can establish its own recovery routine. However, before a user-written file handler returns control to the transfer program, it must drop its own recovery routine, otherwise the transfer program’s abnormal termination processing cannot work. For information about this processing, see the Installation, Operation, and Administration book for your transfer program.
During each function call, the user-written file handler can pass messages to the transfer program for printing or display. For more information about this feature, see Chapter 10, “Issuing Messages from User-Exit Routines” on page 103.

For debugging purposes, the transfer program can trace the user-written file handler. For more information on tracing, see Chapter 11, “Using the Trace Facility” on page 107.

Coding the OPEN Function

The OPEN function does the following:

- Prepares the file for the I/O operations that your user-written file handler performs during GET or PUT function calls. For some access methods, a user-written file handler must issue an explicit macroinstruction to open the file.
- Retrieves additional information that the transfer program needs for a file transfer.

The transfer program requests the OPEN function by placing the character string “OPEN “, in the FUECURCD field of the user file-handler parameter list. The transfer program calls the OPEN function only if it performed its initialization processing successfully.

The OPEN function is always the first function that the transfer program calls. It is called only once during a file transfer.

Using the I/O Area with the OPEN Function

The fifth item in the CALL argument list and the FIOWAD field in the user file-handler parameter list normally contain the address of the I/O area. The user-written file handler uses the I/O area to pass records to the transfer program. However, when the transfer program calls the user-written file handler for the OPEN function, this address contains binary zeros.

The transfer program acquires storage for an I/O area after it regains control from the user-written file handler OPEN function. For this purpose a user-written file handler must write an I/O area length in the FUEIOALN field of the user file-handler parameter list. The I/O area must be large enough to contain the longest record that the transfer program sends or receives during the file transfer. The maximum record size is 32767 bytes.

Both the sending and receiving user-written file handlers must specify an I/O-area length. The two I/O areas need not be of equal size, but the longest record that the transfer programs transfer must fit into both the sending and receiving I/O areas. The transfer program terminates the file transfer if it finds a record that does not fit into the I/O area.

Checking Access Authorization

If the originator of a file-transfer request specifies security parameters, the local transfer program validates the specified user ID and password. It does not check that the user ID has access to the file specified in the file-transfer request. However, the local user-written file handler can perform this check.

The transfer program establishes a temporary security environment, using the parameters from the file-transfer request, before it calls the user-written file handler.
The user-written file handler can use appropriate checking macros, such as RACHECK or an equivalent routing macro, to perform its own access checking. If the user-written file handler does not perform access checking, the file transfer could terminate abnormally. This happens if an access violation is detected by the security system during the OPEN function processing.

If the security system detects an access violation during the OPEN function, the file transfer may ABEND abnormally.

After the OPEN processing has been performed the transfer program removes the temporary security environment. File handler processing for other servers requiring their own security environment must wait.

If a period of 129 seconds elapses and the user-written file handler has not returned from the OPEN function, the user-written file handler is deleted from storage and the security environment is removed. For this reason do not load the user-written file handler into SVA. After this time interval elapses NetView FTP issues message DVG651I.

Return Codes from the OPEN Function

The transfer program expects a return code in the FUERETCD field of the user file-handler parameter list. The following are return codes from the OPEN function:

- X‘00’ OPEN was successful.
- X‘40’ OPEN was unsuccessful.

The transfer program ignores all other return codes and terminates the transfer with an error message.

Coding the GET Function

The transfer program calls this function only on the sending node, to request records from the user-written file handler. NetView FTP optionally compacts, compresses, and encrypts the records and transfers the records to the receiving node.

The user-written file handler must write each record into the I/O area and store its length in the user file-handler parameter list.

The GET function also initiates the normal end of a file transfer. When the sending user-written file handler notifies its transfer program, by passing return code X‘01’, that there are no more records to transfer, the transfer program ends the file transfer.

The sending transfer program requests the GET function by placing the character string “GET ” in the FUECURCD field of the user file-handler parameter list.
The sending transfer program calls the GET function only if it performed the OPEN function successfully. It continues to call the GET function for records as long as no error occurs and the GET function does not notify the transfer program that the sending data set has reached end-of-file. After the last call for the GET function, the transfer program calls the CLOSE function.

Applies to LU 0 Conversations Only

If the transfer program performs an automatic session recovery, the POINT function can interrupt a series of GET function requests.

End of Applies to LU 0 Conversations Only

For a restart from checkpoint with a file-transfer restart, the sending transfer program calls the POINT function before it calls the GET function for the first time. This is to position the sending file correctly.

Using the I/O Area with the GET Function
During the OPEN function the sending user-written file handler requests an I/O area by specifying a value for FUEIOALN. The transfer program acquires storage for this I/O area. The FIOWAD field in the user file-handler parameter list and the fifth word in the CALL argument list point to this I/O area. During each GET function call, the user-written file handler moves one record into this area.

The user-written file handler must ensure that the records written into this I/O area do not exceed its length. The I/O area length is contained in the FIOWLN field of the user file-handler parameter list.

When the transfer program gets storage for the I/O area it initializes this area to X’00’. The transfer program does not reinitialize the I/O area after every GET function.

After every GET function, the FUERECLN field of the user file-handler parameter list must contain the length of the record moved to the I/O area. Only the GET function of the user-written file handler can use this field as an output field.

The transfer program verifies that this field contains a valid value. The value must be positive and less than or equal to the length of the I/O area. If this verification fails, the transfer program issues an error message and terminates the file-transfer step.

Return Codes from the GET Function
The transfer program expects a return code in the FUERETCD field of the user file-handler parameter list. The following GET function return codes are valid:

X’00’ GET was successful.

X’01’ GET has not supplied a record because there are no more records to transfer from the sending file.

X’40’ GET was unsuccessful.

For any other return code from the GET function, the transfer program terminates the transfer with an error message.
The return code X'01' requests the transfer program to initiate the normal end of transfer. It does not expect a record in the I/O area; if the user-written file handler supplies one, the transfer program ignores it.

If a user-written file handler processes two or more files during one file transfer, it must issue X'01' only when the last file has reached end-of-file.

**Coding the PUT Function**

The transfer program calls the PUT function only at the receiving system. It uses PUT to pass the records received from the sending transfer program to the receiving user-written file handler. The sending transfer program obtains these records from the sending user-written file handler. The transfer program stores both the address of the record and its length in the user file-handler parameter list.

The receiving transfer program requests the PUT function by placing the character string “PUT ” in the FUECURCD field of the user file-handler parameter list.

The receiving transfer program calls the PUT function only if the OPEN function was successful. It continues to call the PUT function for records as long as no error occurs, and it continues to receive records from the sending transfer program.

```
Applies to LU 0 Conversations Only
```

The receiving transfer program does not call the POINT function for an automatic session recovery. Therefore, it will not interrupt a series of PUT function requests for a POINT function call.

```
End of Applies to LU 0 Conversations Only
```

For a restart from checkpoint with a transfer restart, the receiving transfer program calls the POINT function before it calls the PUT function for the first time. This is to position the receiving file correctly.

After the last call for a PUT function, the receiving transfer program requests a CLOSE function.

**Using the I/O Area with the PUT Function**

The FIOWAD field in the user file-handler parameter list and the fifth item in the CALL argument list contain the address of the I/O area. When the receiving transfer program calls a PUT function, the I/O area contains a record received from the sending transfer program. If a received record is too long to fit into the I/O area, the receiving transfer program issues an error message and terminates without calling a PUT function for this record. The length of the I/O area is specified during the OPEN function, and must be large enough for the longest record expected during a file transfer.

The FUERECLN field of the user file-handler parameter list contains the length of the record in the I/O area as passed by the GET function of the user-written file handler on the sending node.
Return Codes from the PUT Function

The transfer program expects a return code in the FUERETCD field of the user file-handler parameter list. The following are return codes from the PUT function:

X'00'    PUT was successful.
X'40'    PUT was unsuccessful.

For any other return code from the GET function, the transfer program terminates the file transfer with an error message.

If an unsuccessful PUT function (return code X'40') occurs, the transfer program terminates the file transfer and does not write a checkpoint record, even if the user-written file handler requested one.

Coding the POINT Function

The POINT function is necessary when the file handler carries out a restart from checkpoint. The POINT function positions the file according to the checkpoint information from the checkpoint file. This information is provided by the PUT function and the transfer program gives it to the POINT function.

The POINT function must be able to use the checkpoint information. In particular, the data pointed to by the checkpoint information must be physically present in the sending and receiving files.

The transfer program requests the POINT function by placing the character string “POINT” in the FUECURCD field of the file-handler parameter list.

Applies to LU 0 Conversations Only

For an automatic session recovery, only the transfer program at the sending system calls the user-written file handler for a POINT function. The call can occur at any time between the first GET function and the CLOSE function.

End of Applies to LU 0 Conversations Only

For a restart from checkpoint, the transfer programs in both the sending and the receiving systems call the POINT function. The call occurs immediately after the OPEN call.

Applies to LU 0 Conversations Only

The FUEPOINT field in the file-handler parameter list shows whether the transfer program called the POINT function during an automatic session recovery, or for restart from checkpoint during an automatic or manual transfer restart. In the first case, which can happen only on the sending system, FUEPOINT contains an “R”. In the second case, it contains a “U”.

End of Applies to LU 0 Conversations Only
To indicate a restart case, the FUEPOINT field in the file-handler parameter list contains a “U”.

Notes:
1. In case of an automatic or manual transfer restart the transfer programs can perform a restart from checkpoint only if they can find a checkpoint record on the checkpoint file. Otherwise, the file transfer starts from the beginning of the file and the POINT function is not called.
2. In the case of automatic restart, the transfer program at the sending system can call the POINT function although no checkpoint information is present. In this case, the checkpoint information contains its initial value, X'00'.

Using the I/O Area with the POINT Function
The I/O area is not relevant for the POINT function.

Return Codes from the POINT Function
The transfer program expects a return code in the FUERETCD field of the user file-handler parameter list. The following are return codes for the POINT function:

- X'00': POINT was successful.
- X'01': POINT was unsuccessful because the record identified by the checkpoint information is not on the file.
- X'40': POINT was unsuccessful for some other reason.

For any other return code, the transfer program terminates the transfer with an error message.

When the transfer program encounters a nonzero return code, it terminates the transfer. In addition, for return code X'01', the transfer program issues a special message.

Coding the CLOSE Function
The CLOSE function performs cleanup-processing at the end of the file transfer. For some access methods you must issue a macroinstruction to close the file. The CLOSE function also lets you print information in addition to the transfer program’s file-transfer statistics.

For a file transfer with a user-written file handler, the transfer program does not print either statistical messages, which contain information about the file being transferred, or the number of transferred records. If you want such information, set up the messages during the CLOSE function and pass them to the transfer program. For more information see Chapter 10, “Issuing Messages from User-Exit Routines” on page 103.

Details of the previously processed command and its return code are contained in the FUEPRECD and FUEPRERC fields of the user file-handler parameter list.
If a GET or PUT function error occurs, and you supply a message during the CLOSE function, the transfer program routes the CLOSE function message to the partner node, instead of the PUT or GET function messages.

The transfer program requests the CLOSE function by placing the character string “CLOSE” in the FUECURCD field of the user file-handler parameter list.

The transfer program calls the CLOSE function only if it has previously called the OPEN function, and the return code from it was not X’40’, as shown in Figure 1.

<table>
<thead>
<tr>
<th>Return Code from OPEN Function</th>
<th>CLOSE Function Is Called</th>
</tr>
</thead>
<tbody>
<tr>
<td>X’00’ - OK</td>
<td>Yes</td>
</tr>
<tr>
<td>X’40’ - Not OK</td>
<td>No</td>
</tr>
<tr>
<td>Other - Invalid</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The CLOSE function is the last file-handler function that the transfer program calls before it terminates the transfer, either normally or in error. The transfer program calls the CLOSE function only once for each file transfer.

When the transfer program encounters an abnormal termination and its ESTAE recovery routine (MVS, VM), or its ABEND exit routine (VSE), is scheduled, the routine attempts to call the CLOSE function, unless the return code from the previous OPEN function was X’40’.

However, there can be situations where the routine cannot work as it is designed. For example, if your system terminates abnormally, the transfer program cannot call the CLOSE function.

Note: The transfer program does not call this function during automatic session recovery. If you need to close and reopen a file for automatic session recovery you must use the user-written file handler POINT function.

Using the I/O Area with the CLOSE Function
When the transfer program regains control after the CLOSE function of the file handler has finished, it releases the storage of the I/O area. Any data in the I/O area is irrelevant once the transfer program calls the CLOSE function.

Return Codes from the CLOSE Function
The transfer program expects a return code in the FUERETCD field of the user file-handler parameter list. The following are return codes from the CLOSE function:

- X’00’ CLOSE was successful.
- X’20’ CLOSE was successful but some minor errors occurred.
- X’40’ CLOSE was unsuccessful.
For any other return code, the transfer program terminates the transfer with an error message.

If a file transfer completed without errors up to the CLOSE function, the return code issued by the CLOSE function tells the transfer program whether or not the file transfer was successful.

Return codes X’00′ and X’20′ indicate that the CLOSE function was successful. If no error has occurred, the transfer program assumes that the file transfer was successful. If the transfer program wrote a checkpoint record to the checkpoint file during the file transfer, it deletes it when it receives either of these return codes.

Code X’20′ indicates that the closing process was successful but that a minor error occurred, for example, the user-written file handler could not release the storage area it obtained during one of the previous function calls. In its final return code for the transfer, the transfer program indicates a warning condition. Your user-written file handler should pass error messages to the transfer program, to provide information about the nature of any minor errors. See Chapter 10, “Issuing Messages from User-Exit Routines” on page 103 for more information.

The transfer program treats a return code of X’40′ at the sending node as if it were a return code of X’20′. The transfer program does not consider a CLOSE function error on the sending file to be a serious error.

The transfer program assumes that a return code of X’40′ from the receiving node is an indication that the file transfer was not successful because the user-written file handler could not successfully close the receiving data set. When this happens, the transfer program does not delete any previously written checkpoint records. It routes an error message, which is either supplied by the user-written file handler or written by the transfer program, to the sending node, and indicates in its final return code that the file transfer was not successful.

### Designing a User-Written File Handler without Checkpointing

If you are not going to do checkpointing in the user-written file handler, do not set the FUECRREQ field to “C” during a PUT function (see FUECRREQ in “Using the User File-Handler Parameter List” on page 23). This ensures that you never request the transfer program to write a checkpoint record to the checkpoint file.

---------- Applies to LU 0 Conversations Only ----------

Set FUERECOV to “N” in your OPEN function (see FUERECOV in “Using the User File-Handler Parameter List” on page 23), so preventing the transfer program from performing an automatic session recovery.

---------- End of Applies to LU 0 Conversations Only ----------

Your user-written file handler can still do file transfer restarts. A file transfer, however, is always restarted from the beginning of the data set.
Designing a User-Written File Handler with Checkpointing

The following will help you to design checkpoint support within a user-written file handler. For further information about the checkpoint facility, refer to the Installation, Operation, and Administration book for your transfer program.

A user-written file handler can carry out a restart on an interrupted file transfer. The sending user-written file handler must have the same checkpoint capabilities as the receiving user-written file handler.

Periodically during a file transfer, a receiving server can record how much of a file has been transferred successfully by writing a checkpoint record to the checkpoint file. This is called taking a checkpoint.

If a file transfer is interrupted and then resumed, the receiving server informs the sending server about where in the file it last took a checkpoint. The sending server can then continue from the last checkpoint instead of transferring the whole file from the beginning.

A checkpoint record consists of a key and several information fields. The key identifies the file transfer to which the checkpoint record belongs. This ensures that the receiving transfer program does not retrieve checkpoint information that belongs to a different transfer.

The information fields contain information about the file transfer. These fields are used by the file handlers to position the sending and receiving files so that the file transfer can be continued.

Applies to NetView FTP V2.2 MVS Only

The request originator can specify now whether a file transfer is to be restarted from a recorded checkpoint, or from the beginning of the source data set. The value of the Restart Point parameter is passed to the user-written file handler.

End of Applies to NetView FTP V2.2 MVS Only

Checkpointing for the OPEN Function

Applies to LU 0 Conversations Only

In the OPEN function, you establish whether or not you want automatic session recovery.

You can prevent the transfer program from implementing automatic session recovery by assigning “N” to the FUERECOV field of the user file-handler parameter list. Any other entry into this field, including a blank, signifies that you want automatic session recovery. The transfer program and the user-written file handler cannot change this value during execution of the other functions.

End of Applies to LU 0 Conversations Only
If the FUEKNFTP field contains the character “Y” on invocation of the OPEN function, NetView FTP already generated the checkpoint record key. In this case, the user-written file handler should not overwrite the checkpoint record key area. NetView FTP ignores data that is written into this area under these circumstances. If the checkpoint record key is not supplied by NetView FTP, the following applies.

In the OPEN function, the user-written file handler establishes the data for the key of the checkpoint record.

The third item in the CALL argument list and the FUEKEYP field of the user file-handler parameter list contain the address of an 88-byte area. This is the checkpoint record key area.

The transfer program initializes the checkpoint key area with hexadecimal zeros. The sending and receiving user-written file handlers can fill this area during the OPEN function.

“Contents of the Checkpoint Record Key Provided by User-Written File Handlers” on page 17 shows how the transfer program uses the supplied data to form the checkpoint record key.

Checkpointing for the PUT Function

In the PUT function you supply information to the two checkpoint information fields. You can also request the transfer program to write the current checkpoint record from its storage area to the checkpoint file.

For each successful call by the transfer program, the PUT function must supply the checkpoint information. The transfer program then inserts this information in the checkpoint record that it keeps in its own storage area.

The addresses of the two data fields for checkpoint information are contained in the user file-handler parameter list (FUECRPI, FUECRPO) and in the sixth and seventh fullwords of the CALL argument list. Each field has a length of 255 bytes. The storage for each field is allocated by the transfer program.

When the transfer program requests the first PUT function of a file transfer, it initializes the two checkpoint information fields with X’00’. The transfer program does not reinitialize the two fields after every PUT function. The fields remain in the state created by the previous PUT function. You can use one or both data fields.

Use one field for the checkpoint information of the sending file and the other field for the checkpoint information of the receiving file. However, both fields must be supplied by the PUT function at the receiving system.

After each PUT function, the transfer program checks the FUECRREQ field of the file-handler parameter list. If the user-written file handler sets this field to “C”, the transfer program writes the current checkpoint record to the checkpoint file.

The transfer program stores the most recent checkpoint information in the checkpoint record before writing the record to the checkpoint file.

If the user-written file handler specifies a value for FUECRREQ during a function other than the PUT function, the transfer program ignores it but does not reset it to blank.
For more information about when to write a checkpoint record to the checkpoint file refer to “Writing a Checkpoint Record to the Checkpoint File” on page 18.

**Specifying Checkpoint Record Key Data**

If FUEKNFTP does not contain the character “Y”, the checkpoint record key is not supplied by NetView FTP, and the following applies.

During the OPEN function, the transfer program requests the user-written file handlers in the sending and receiving systems to specify the checkpoint record key. This key is used only to identify the record in the checkpoint file. If checkpointing is not being used, then it is not necessary to specify the key.

Automatic session recovery does not require checkpointing.

The receiving transfer program creates the key by using checkpoint key data that the sending and the receiving user-written file handlers supply during their OPEN functions. This information makes up the checkpoint record key for this file transfer.

Pay special attention to the following when you create a key:

- The checkpoint key data from the file handlers should contain file-dependent information. For example, the information could be file names that are unique to a system. Alternatively, the information could be a special code derived from the parameters in the file-transfer request, where the code uniquely identifies the file transfer.

- The transfer programs do not check the information provided by the user-written file handler. They do not impose any restriction on what the user-written file handler provides, except for the length of the fields.

- When a file transfer restarts, the OPEN function must provide the same checkpoint key data as it provided for the initial start of the file transfer. Otherwise, the transfer programs cannot carry out restart from checkpoint because they cannot find the previously written checkpoint record.
Contents of the Checkpoint Record Key Provided by User-Written File Handlers

Figure 2 shows the contents of the key of a checkpoint record provided by user-written file handlers.

| Checkpoint key information from the sending user-written file handler | 88 characters |
| (Reserved) | 31 characters |
| Identifier for sending program (supplied by transfer program) | 8 characters |
| Checkpoint key information from the receiving user-written file handler | 88 characters |
| (Reserved) | 31 characters |
| Identifier for receiving program (supplied by transfer program) | 8 characters |

Figure 2. Checkpoint Record Key when Created in User-Written File Handlers

Specifying Checkpoint Record Information

Two fields in the checkpoint record contain information about how far the file transfer had proceeded when the PUT function of a file handler puts this information into the checkpoint record. The POINT function of a file handler uses this information to position a file for a restart from checkpoint.

The receiving transfer program transfers the restart information for the sending file, which is contained in the field FUECRPI, to the sending transfer program so that both have the same checkpoint information, although the information is created only by the PUT function of the receiving file handler.

If a file transfer with a user-written file handler does not use checkpointing, the user-written file handler does not have to supply any checkpoint information.

When you supply or retrieve checkpoint information, consider the following:

- The transfer program does not check the contents of these fields. There are no restrictions on the kind of information you can put in each field.
- You must code the user-written file handlers so that each file handler understands the data that the receiving file handler has supplied. In most cases the checkpoint information is identical for the sending and the receiving data set. For example, you can record the status of a file transfer by a record count. If the information is not identical, the receiving file handler must know how to retrieve the checkpoint information for the sending file.
- When the transfer program calls the POINT function, it passes the checkpoint information fields as the PUT function has supplied them.
- At the sending side, only those 255 bytes of information FUECRPI points to are available.
Writing a Checkpoint Record to the Checkpoint File

A receiving user-written file handler that updates the checkpoint information and then requests the transfer program to write the checkpoint record to the checkpoint file must observe the following:

- The access method used on the sending and the receiving systems determines whether both user-written file handlers can carry out a POINT function using identical checkpoint information, or whether they need different information. For example, the sending user-written file handler might need the key of the record while the receiving user-written file handler needs the record number.

- The checkpoint information for the sender might be different from that for the receiver. You must ensure that, after the transfer program has called the POINT function, the positions in the sending and receiving data sets are logically the same. The user-written file handler in the receiving system determines the use of the two data fields for checkpoint information. The sending user-written file handler must be able to understand the information it receives.

- When you want the receiving transfer program to write a checkpoint record, the checkpoint position in the receiving data set should reflect the physical state of the receiving data set. For instance, if your access method carries out buffering, make sure that you write the buffers out to the physical device. Otherwise, with a restart from checkpoint, the POINT function will not be able to find the position, in the receiving data set, that is identified by the checkpoint information.

To carry out regular periodic writing of the checkpoint records to the checkpoint file, you can use the value of the server initialization parameter RUNUMCR. The values of these parameters define the frequency with which the receiving transfer program periodically writes a checkpoint record to the checkpoint file when it uses its own file handlers.

If you specify BYTECR, the appropriate value for RUNUMCR is used. If you set all of these server initialization parameters to 0, no regular checkpointing is carried out. For more information refer to Installation, Operation, and Administration for your NetView FTP.

The following information indicates the frequency of regular checkpoints:

<table>
<thead>
<tr>
<th>Applies to LU 0 Conversations Only</th>
</tr>
</thead>
<tbody>
<tr>
<td>FUERUNCR</td>
</tr>
<tr>
<td>FUECPIBC</td>
</tr>
<tr>
<td>FUECPITS</td>
</tr>
</tbody>
</table>

End of Applies to LU 0 Conversations Only
Implementing Restart from Checkpoint for an Automatic Session Recovery (LU 0)

During the OPEN function, the user-written file handler establishes whether or not it supports automatic session recovery. This is explained in “Checkpointing for the OPEN Function” on page 14.

When automatic session recovery is specified in the user-written file handler, the transfer program does not try to establish a new session immediately, but behaves as specified in the file-transfer request’s automatic transfer restart parameter. In this way, if automatic session recovery cannot be performed, automatic transfer restart is still possible.

The transfer program uses the checkpoint record that it keeps in its storage area when it performs an automatic session recovery. It does not use the checkpoint record in the checkpoint file. An automatic session recovery always uses a restart from checkpoint. However, only the sending data set needs to be repositioned.

The following describes the general flow of events for an automatic session recovery:

1. The receiving transfer program keeps and updates the checkpoint record in its own storage area. After a successful PUT function, the receiving user-written file handler updates the checkpoint information. This ensures that the checkpoint record reflects the current state of the file transfer. Taking a record count is the simplest way of recording such information.

   The checkpoint information for the receiving file must, therefore, reflect the state of the receiving file after a PUT function. For the sending file it must indicate which record has just been written to the receiving file.

2. When a session interruption occurs, the transfer programs attempt to establish a new session. If they succeed, the receiving transfer program sends its current checkpoint record to the sending transfer program.

3. The sending transfer program calls the POINT function of the user-written file handler.

   If records have been lost, the sending file handler must take up a new position in the sending file. The POINT function determines, from the checkpoint information, how much of the sending file the receiving user-written file handler had received when the interrupt occurred.

   The sending transfer program calls the POINT function. It does not tell the user-written file handler in the receiving system that it is doing an automatic session recovery.

4. With the next GET function, the sending file handler supplies the first record that the receiving file handler failed to write to the receiving file.

   The sending user-written file handler then transfers the remaining records of the sending file.

See “Sequence of Function Calls for a User-Written File Handler” on page 34 for the sequence of function calls for automatic restart.
Example of Automatic Session Recovery
The following example describes the sequence of events involved in a typical automatic session recovery:

1. The receiving transfer program writes the 1000th record to the receiving data set.
2. The receiving user-written file handler updates the checkpoint information. Somewhere in the two 255-byte fields, it stores the number of records received (1000).
3. A temporary session break causes an automatic session recovery.
4. The receiving transfer program informs the sending transfer program about the session break. By this time, the sending transfer program has processed the 1050th record. So, 50 records are “lost.”
5. The receiving transfer program sends the checkpoint record, containing the number of records received (1000), to the sending transfer program.
6. With no intervening CLOSE function or OPEN function, the sending transfer program calls the POINT function, and passes the checkpoint information to the sending user-written file handler.
7. The sending user-written file handler uses the checkpoint data to determine that there are 1000 records in the receiving data set. As record number 1001 is the first one missing, it positions itself at record 1001.
8. With the next GET function, the sending user-written file handler passes record 1001 to the sending transfer program, which then resumes the file transfer.

Note: The receiving user-written file handler was not involved in, and was not aware that the transfer program had carried out, the automatic session recovery.

Implementing Restart from Checkpoint for a Transfer Restart
Refer to Installation, Operation, and Administration for your transfer program to see the types of errors that can be handled by the file-transfer restart facilities. Restart from checkpoint can be used with both automatic transfer restart and manual transfer restart, and the sequence of events for such restarts is identical.

The transfer program carries out a restart from checkpoint only if the following conditions are met:

- The receiving transfer program can find a checkpoint record in the checkpoint file when it performs a transfer restart.
- For a manual transfer restart, the value of the Restart from Checkpoint parameter in the file-transfer request is YES. See the User’s Guide for your transfer program for a description of the Restart from Checkpoint parameter.

Does Not Apply to NetView FTP V2.2 MVS

End of Does Not Apply to NetView FTP V2.2 MVS
The value of the Restart Point parameter in the file-transfer request is CHKPT.

If the value of the Restart Point parameter in the file-transfer request is BEGIN, NetView FTP restarts a file transfer from the beginning of the sending file. A user-written file handler can implement processing for this type of restart. The value of the Restart Point parameter is provided to the user-written file handler in the FUERSTPT field.

See the NetView FTP Parameter Reference for a description of the Restart Point parameter.

If the transfer is to restart from a checkpoint, the receiving user-written file handler must have requested that a checkpoint record be written to the checkpoint file.

The following describes the events that take place during a transfer restart from checkpoint:

1. The receiving transfer program keeps and updates the checkpoint record in its own storage area. When the receiving user-written file handler carries out a successful PUT function, it updates the checkpoint information. This ensures that the checkpoint record reflects the current state of the file transfer. A record count is the simplest way of recording such information.

   The checkpoint information for the receiving file must reflect the state of the receiving file after the current PUT function. For the sending file, it must denote the record that has just been written to the receiving file.

2. The receiving transfer program writes a checkpoint record to the checkpoint file only when requested to by the PUT function of the receiving user-written file handler. The transfer program allows the user-written file handler the freedom to request or not to request the writing of the checkpoint record.

   Within the PUT function, enter a “C” in the FUECRREQ field of the user file-handler parameter list. This tells the transfer program to write a checkpoint record to the checkpoint file. If the PUT function was not successful, the transfer program does not write the checkpoint record, even if the user-written file handler requested it.

   For details of what you have to take into account when you request the writing of a checkpoint record read “Writing a Checkpoint Record to the Checkpoint File” on page 18.

3. If, after an interrupt in the file transfer, the file-transfer request is processed again, the receiving transfer program searches the checkpoint file for a previously written checkpoint record. If it finds one, it looks into the file-transfer request to see if restart from checkpoint is required. This is always true with automatic transfer restart. With manual transfer restart, it depends on what the originator of the file-transfer request specified. If a restart from checkpoint can be performed, the receiving transfer program sends the checkpoint record to the sending transfer program. Both transfer programs then call the OPEN function and then the POINT function.
4. When the transfer programs call the POINT function, they pass the checkpoint information to the user-written file handlers.

5. The user-written file handlers position their data sets and the transfer programs resume the file transfer with the next GET and PUT function calls.

See “Sequence of Function Calls for a User-Written File Handler” on page 34 for the sequence of function calls for a restart from checkpoint with an automatic transfer restart or manual transfer restart.

Example of Restart from Checkpoint during a Transfer Restart

The following example illustrates a restart from checkpoint during a transfer restart:

1. The receiving transfer program passes the 1000th record to the PUT function of the receiving file handler.

2. The receiving user-written file handler updates the checkpoint record and requests its transfer program to write the checkpoint record to the checkpoint file.

3. The receiving transfer program passes the 1050th record to the PUT function.

4. A telecommunication error occurs and the session cannot be reestablished. At this time the sending transfer program has sent 1100 records.

5. The file-transfer request is requeued if a transfer restart was requested.

6. Sometime later the request is rescheduled and the session is reestablished.

7. Both transfer programs call the OPEN functions of the two user-written file handlers. Either NetView FTP or these functions supply the checkpoint key data, which enables the receiving transfer program to find the most recent checkpoint record.

8. The receiving transfer program retrieves the checkpoint record and passes it to the POINT functions of the two file handlers.

9. The two user-written file handlers position their data sets ahead of record 1001.

10. With the next GET function, the sending user-written file handler passes record 1001 to the sending transfer program.

11. The receiving transfer program passes record 1001 to the PUT function. Because this record exists already in the receiving file, the receiving file handler must decide whether to discard the record or to overwrite the existing record with the newly received one. This decision can depend on the access method used.
Using the User-Written File-Handler Interfaces

The interface between the transfer program and the user-written file handler consists of:

- The user file-handler parameter list.
- A list of addresses that the transfer program passes to the file handler each time it calls it. This is called the CALL argument list.

Using the User File-Handler Parameter List

The user file-handler parameter list is the main part of the interface between the user-written file handler and the transfer program. Upon each call of the file handler, the first word of the CALL argument list addressed by Register 1 points to the user file-handler parameter list.

An assembler dummy section of the parameter list is available with its corresponding macro, DVG$UPL. If you use a higher-level programming language to write a user-written file handler, you must define a structure for this parameter list. The offsets are shown in Figure 3 on page 25, and Figure 4 on page 26.

Symbolic names address the fields in the user file-handler parameter list. The assembler dummy section uses these symbolic names. If you write a structure for the parameter list in the user-written file handler, you can use different names for the fields. You must use the fields consistently, however, and give the corresponding fields the same length as shown in the layout description for this parameter list.

The parameter list consists of three main parts for NetView FTP V2.2 MVS and two main parts for NetView FTP VM and NetView FTP VSE. The first part contains the read-only section from which the file handler can retrieve the required information during processing. The transfer program does not allow the file handler to overwrite this information.

The second part contains the read/write section. After every function call, the file handler must fill some fields of this section. The file handler can set others, optionally, during specific stages of the file transfer. Figure 3 on page 25 indicates which parameters are mandatory and which are optional.

The third part contains the extensions made for NetView FTP V2.2 MVS.

Layout of the User File-Handler Parameter List

Figure 3 shows the layout of the read-only part of the file-handler parameter list; Figure 4 shows the layout of the read/write part; Figure 5 shows the layout of the additional read-only part for NetView FTP V2.2 MVS. The abbreviations in the columns to the right (under “File-Handler Function”) have the following meanings:

I Input field, NetView FTP provides information to the user-written file handler.
N Do not use the value for the indicated function.
R Required output field, the user-written file handler provides information to NetView FTP.
O Optional output field, the user-written file handler provides information to NetView FTP.
<table>
<thead>
<tr>
<th>Displacement (Offset)</th>
<th>Length</th>
<th>Symbolic Name</th>
<th>Description</th>
<th>File-Handler Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>FUEXPLRO</td>
<td>Read-only part of parameter list</td>
<td>OPEN CLOSE PUT GET POINT</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>*</td>
<td>Reserved</td>
<td>I I I I I</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>FUEINPTR</td>
<td>Pointer to information in UINF0 to UINF9</td>
<td>I I I I I</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>FUERUCNT</td>
<td>Current request unit counter (LU 0 only)</td>
<td>I</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>FUERUNCR</td>
<td>Frequency of request checkpointing (LU 0 only)</td>
<td>I</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>FMODE</td>
<td>Transfer mode: 'I'= Program is sender 'O'= Program is receiver</td>
<td>I I I I I</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>FUEPOINT</td>
<td>Situation when point is called: 'R'= Automatic session recovery (LU 0 only) 'U'= Automatic transfer restart or manual transfer restart</td>
<td>I</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>FIOWLN</td>
<td>I/O area length</td>
<td>N I I</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>FIOWAD</td>
<td>I/O area address</td>
<td>N I I</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>*</td>
<td>Reserved</td>
<td>I</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>FUEPHN</td>
<td>User-written file handler module name</td>
<td>I</td>
</tr>
<tr>
<td>36</td>
<td>24</td>
<td>FUECRPI</td>
<td>Pointer to checkpoint information (first field)</td>
<td>I I</td>
</tr>
<tr>
<td>40</td>
<td>28</td>
<td>FUECRPO</td>
<td>Pointer to checkpoint information (second field)</td>
<td>I I</td>
</tr>
<tr>
<td>44</td>
<td>2C</td>
<td>FUEKEYP</td>
<td>Pointer to area for checkpoint-record key information</td>
<td>I</td>
</tr>
<tr>
<td>48</td>
<td>30</td>
<td>FUEMSGP</td>
<td>Pointer to message area</td>
<td>I I I I I</td>
</tr>
<tr>
<td>52</td>
<td>34</td>
<td>FUECURCD</td>
<td>Current command</td>
<td>I I I I I</td>
</tr>
<tr>
<td>57</td>
<td>39</td>
<td>*</td>
<td>Reserved</td>
<td>I I I I I</td>
</tr>
<tr>
<td>60</td>
<td>3C</td>
<td>FUEPRED</td>
<td>Previous command</td>
<td>I I I I I</td>
</tr>
<tr>
<td>65</td>
<td>41</td>
<td>*</td>
<td>Reserved</td>
<td>I I I I I</td>
</tr>
<tr>
<td>68</td>
<td>44</td>
<td>FUEPRERC</td>
<td>Return code from previous command</td>
<td>I I I I I</td>
</tr>
</tbody>
</table>
### Figure 4. The File-Handler Parameter List (Read/Write Part)

<table>
<thead>
<tr>
<th>Displacement (Offset)</th>
<th>Length</th>
<th>Symbolic Name</th>
<th>Description</th>
<th>File-Handler Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>Hex</td>
<td></td>
<td></td>
<td>OPEN</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>28</td>
<td>FUEXPLRW</td>
<td>Read/Write part of parameter list</td>
</tr>
<tr>
<td>72</td>
<td>48</td>
<td>4</td>
<td>FUERETCD</td>
<td>Return code from user file handler</td>
</tr>
<tr>
<td>76</td>
<td>4C</td>
<td>4</td>
<td>FUEIOALN</td>
<td>I/O area length</td>
</tr>
<tr>
<td>80</td>
<td>50</td>
<td>4</td>
<td>FUETRCP</td>
<td>Pointer to area to be traced</td>
</tr>
<tr>
<td>84</td>
<td>54</td>
<td>4</td>
<td>FUETRCL</td>
<td>Length of area to be traced</td>
</tr>
<tr>
<td>88</td>
<td>58</td>
<td>4</td>
<td>FUERECLN</td>
<td>Record length</td>
</tr>
<tr>
<td>92</td>
<td>5C</td>
<td>1</td>
<td>FUECRREQ</td>
<td>Request a checkpoint record to be written after a PUT: 'C' = Write a checkpoint record</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Request = Do not write a checkpoint record</td>
<td></td>
</tr>
<tr>
<td>93</td>
<td>5D</td>
<td>1</td>
<td>LU 0: FURECOV</td>
<td>Automatic session recovery indicator: 'N' = Automatic session recovery is not allowed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Other = Automatic session recovery is allowed</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>94</td>
<td>5E</td>
<td>6</td>
<td>*</td>
<td>Reserved</td>
</tr>
<tr>
<td>100</td>
<td>64</td>
<td>32</td>
<td>FUWKAREA</td>
<td>Can be used as a work area for a user-written file handler</td>
</tr>
</tbody>
</table>

Chapter 2. Writing a User-Written File Handler  25
### Fields in the User File-Handler Parameter List

**FUEXPLRO**

The label of the read-only part of the user file-handler parameter list. The user-written file handler must not modify this part of the parameter list.

**FUEINPTR**

Points to the area that contains the parameter values that the originator of the file-transfer request specifies for special use with user-written file handlers. The structure of this area is explained in “Presenting Information Parameters to the User-Written File Handler” on page 32.

---

**FUERUCNT**

Contains the count of RUs sent or received.

**FUERUNCR**

Contains the value for the frequency of regular checkpointing. It is an initialization parameter value of the server on which the file transfer runs.

---

**FMODE**

Contains the transfer mode of the local transfer program, either “I” for “input” (has to support the GET function) or “O” for “output” (has to support the PUT function). If a file handler can run on the sending node and on the receiving node, use this field for transfer-mode-dependent processing.

---

<table>
<thead>
<tr>
<th>Displacement (Offset)</th>
<th>Length</th>
<th>Symbolic Name</th>
<th>Description</th>
<th>File-Handler Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decimal</td>
<td>Hex</td>
<td>Symbolic Name</td>
<td>Description</td>
<td>OPEN</td>
</tr>
<tr>
<td>132</td>
<td>84</td>
<td>28</td>
<td>FUERO02</td>
<td>Read-only part of parameter list</td>
</tr>
<tr>
<td>132</td>
<td>84</td>
<td>1</td>
<td>FUEKFTP</td>
<td>Checkpoint record key indicator: ‘Y’= Checkpoint record key provided by NetView FTP</td>
</tr>
<tr>
<td>133</td>
<td>85</td>
<td>1</td>
<td>FUERSTPT</td>
<td>Restart point indicator: ‘C’= Restart from checkpoint ‘B’= Restart from beginning of the sending file</td>
</tr>
<tr>
<td>134</td>
<td>86</td>
<td>2</td>
<td>*</td>
<td>Reserved</td>
</tr>
<tr>
<td>136</td>
<td>88</td>
<td>4</td>
<td>FUECPIBC</td>
<td>Checkpoint interval in bytes</td>
</tr>
<tr>
<td>140</td>
<td>8C</td>
<td>4</td>
<td>FUECPITS</td>
<td>Checkpoint interval in seconds</td>
</tr>
<tr>
<td>144</td>
<td>90</td>
<td>16</td>
<td>*</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

---

Applies to LU 0 Conversations Only.
FUEPOINT
Indicates whether or not the transfer program calls a POINT function for an automatic session recovery (LU 0 conversations only), an automatic transfer restart, or a manual transfer restart.

For automatic session recovery, this field contains “R” (LU 0 conversations only). For automatic transfer restart or manual transfer restart, this field contains “U”.

The transfer program resets FUEPOINT after it has called the user-written file handler for the POINT function.

FIOWLN
Identifies the length of the I/O area. The longest record that the user-written file handler passes to (GET function) or expects from (PUT function) the transfer program to be transferred must fit into this I/O area.

FIOWAD
Contains the address of the I/O area that the transfer program uses for all function calls after the OPEN function. The transfer program gets storage for this I/O area when it regains control after the OPEN function in the user-written file handler. The fifth word in the CALL argument list contains the same value.

For reasons that are internal to the transfer program, the I/O area address in FIOWAD can be different for each call of the user-written file handler. Your program must always use the currently supplied address.

FUEPHN
Contains the name of the user-written file handler that the transfer program loaded.

FUECRPI
Contains the address of the first of two 255-byte areas for checkpoint information. The user-written file handler must fill this area during execution of the PUT function if it is to carry out checkpointing. This area contains the restart information for the sending file. The information is made available to the sending system. The area serves as an output field for the PUT function, that is, on the receiving side, and as an input field for the POINT function on the sending side. The sixth word of the CALL argument list contains the same value.

FUECRPO
Contains the address of the second of two 255-byte areas for checkpoint information. The user-written file handler must fill this area during execution of the PUT function if it is to carry out checkpointing. This area contains the restart information for the receiving file. The information is not made available to the sending system. The area serves as an output field for the PUT function and as an input field for the POINT function. The seventh word of the CALL argument list contains the same value.

FUEKEYP
Points to an 88-byte area for the key of the checkpoint record. If the routines implement checkpointing and NetView FTP has not already provided the key of the checkpoint record, the sending and receiving user-written file handlers must fill this area during the OPEN function. The third word in the CALL argument list contains the same address.
FUEMSGP
This field points to an area of 1844 bytes that is provided by the transfer program. The user-written file handler can fill this area with messages during the execution of a function. The fourth word of the CALL argument list also points to this message area. After the transfer program regains control, it prints the messages that the user-written file handler supplied.

FUECURCD
This field contains the function code that the user-written file handler performs during the current call. The field can contain one of the following values:

"OPEN"
"CLOSE"
"PUT"
"GET"
"POINT"

FUEPRECD
Contains the command that the FUECURCD field contained during the previous call of the user-written file handler. The field can contain one of the following values:

"OPEN"
"PUT"
"GET"
"POINT"

When the transfer program calls the user-written file handler for the first time, this field contains blanks.

FUEPRERC
Contains the return code from the previously performed file-handler function. The transfer program initializes FUEPRERC to X'00' before it calls the user-written file handler for the first time.

FUEXPLRW
Denotes the beginning of the read/write part of the user file-handler parameter list.

FUERETCD
Contains the return code from the current user-written file handler function. The transfer program initializes FUERETCD to X'00' before it calls the user-written file handler.

FUEIOALN
Contains the I/O-area length as specified by the file handler during the OPEN function.

FUETRCP
Contains a pointer to the start of the area that the transfer program is to trace. After taking the trace, the transfer program resets FUETRCP to zero.

If FUETRCP, together with FUETRCL, do not specify a valid storage area, the results are unpredictable.

FUETRCL
Contains the length of the area that the transfer program is to trace.

After taking the trace, the transfer program resets FUETRCL to zero.
FUERECLN
Contains the value of the record length supplied by the file handler in the I/O area during the current GET function.

The transfer program initializes FUERECLN to binary zeros before it calls the OPEN function of the file handler.

FUECRREQ
Contains a value that determines whether or not the transfer program writes a checkpoint record to the checkpoint file after it regains control from a PUT function.

The transfer program writes a checkpoint record to the checkpoint file only when this field contains “C”.

Applies to LU 0 Conversations Only

FUERECOV
Contains the value that determines whether the transfer program performs automatic session recovery. This value is set in the OPEN function.

If the user-written file handler sets the value to “N”, the transfer program does not perform automatic session recovery.

Any value other than “N”, including a blank, indicates that the user-written file handler wants the transfer program to perform automatic session recovery.

Once a value is set, neither the user-written file handler nor the transfer program can change it.

End of Applies to LU 0 Conversations Only

FUWKAREA
This area is not used by the transfer program. It can be used as a work area for a user-written file handler.

Applies to NetView FTP V2.2 MVS Only

FUEKNFTP
Contains a value that indicates whether NetView FTP has generated the checkpoint record key or not. If this field contains “Y”, NetView FTP did so, and the user-written file handler should not provide own information for the checkpoint record key. If the user-written file handler does anyhow, the information is ignored by NetView FTP.

FUERSTPT
Contains a value that indicates to the user-written file handler what was specified for the Restart Point parameter. A value of “C” indicates that a restart from checkpoint was requested, a value of “B” indicates that a restart from the beginning of the sending data set was requested.

FUECPIBC
Contains the value of the server initialization parameter Byte Checkpoint Interval (already converted into bytes). This field denotes the interval between regular checkpoints in bytes.
FUECPITS
Contains the value of the server initialization parameter Time Checkpoint Interval. This field denotes the interval between regular checkpoints in seconds.

End of Applies to NetView FTP V2.2 MVS Only

Using the CALL Argument List
When the transfer program calls a user-written file handler, Register 1 points to a list of seven fullwords that contains the addresses of the arguments passed to the user-written file handler, as shown in Figure 6.

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–4</td>
<td>Address of first argument</td>
</tr>
<tr>
<td>5–8</td>
<td>Address of second argument</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>25–28</td>
<td>Address of seventh argument</td>
</tr>
</tbody>
</table>

*Figure 6. The CALL Argument List*

First Fullword: File-Handler Parameter List
The first fullword in the CALL argument list contains the address of the user file-handler parameter list. This parameter list is described in “Using the User File-Handler Parameter List” on page 23. The file-handler parameter list also contains the addresses of those areas that can be addressed by the other six fullwords. Your file handler can use either the addresses from the user file-handler parameter list or the CALL argument list.

Second Fullword: File-Handler Parameter Area
This fullword contains the same values as the FUEINPTR field of the user file-handler parameter list. See page 27 for a description.

Third Fullword: Checkpoint Record Key Area
This fullword contains the same values as the FUEKEYP field of the user file-handler parameter list. See page 28 for a description.

Fourth Fullword: Message Area
This fullword contains the same values as the FUEMSGP field of the user file-handler parameter list. See page 29 for a description.

Fifth Fullword: Input/Output Area
This fullword contains the same values as the FIOWAD field of the user file-handler parameter list. See page 28 for a description.
Sixth Fullword: Checkpoint Information Area (Sending File)
This fullword contains the same values as the FUECRPI field of the user file-handler parameter list. See page 28 for a description.

Seventh Fullword: Checkpoint Information Area (Receiving File)
This fullword contains the same values as the FUECRPO field of the user file-handler parameter list. See page 28 for a description.

Presenting Information Parameters to the User-Written File Handler
In a file-transfer request a user can specify up to ten parameters for use with user-written file handlers. These are called information parameters. For information about how to specify these parameters refer to the User's Guide for NetView FTP VM or NetView FTP VSE or to the NetView FTP Parameter Reference for NetView FTP V2.2 MVS.

When the transfer program calls a user-written file handler, it passes the address of a 660-byte storage area that contains the values of the ten parameters.

Note: If the values for the parameters are specified in quotes, the batch job interface routine (EXEC interface with NetView FTP VM) removes the quotes, whereas the interactive interface (not available with NetView FTP VSE) and application program interface routines do not remove them.

Figure 7 shows the layout of the storage area:

Bytes

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1–66</td>
<td>Value of first parameter</td>
</tr>
<tr>
<td>67–132</td>
<td>Value of second parameter</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>595–660</td>
<td>Value of tenth parameter</td>
</tr>
</tbody>
</table>

Figure 7. Storage Area for the Information Parameter Values

If fewer than 66 bytes are specified for one parameter, the value is padded to the right with blanks. If fewer than 10 parameter values are specified, the area for the remaining parameters is set to blanks (X'40').
Using the Registers

The linkage between the transfer program and a user-written file handler follows normal linkage conventions. On entry to the file handler, the general registers have the following contents:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>1</td>
<td>Points to a list of seven fullwords (the Call Argument List)</td>
</tr>
<tr>
<td>2 to 12</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>13</td>
<td>Contains the address of a save area of 18 fullwords in the transfer program</td>
</tr>
<tr>
<td>14</td>
<td>Contains the return address to the transfer program</td>
</tr>
<tr>
<td>15</td>
<td>Contains the entry address to the exit routine</td>
</tr>
</tbody>
</table>

Saving the Registers

On entry to a user-written file handler, registers 0 to 12, 14, and 15 must be stored in the save area to which Register 13 points. Use fullwords 4 to 18 for this purpose.

The user-written file handler must also have a save area of 18 fullwords. On entry to a user-written routine, the routine must store Register 13 (the transfer program’s save area address) in the second word of the file handler’s save area. The third word of the transfer program’s save area must contain the address of the file handler’s save area.

After saving the transfer program’s registers, a user-written file handler can use the registers as it likes.

Returning to the Transfer Program

Before a routine returns control to the transfer program, it must reload the transfer program’s registers from the transfer program’s save area. The routine must reset Register 13 to the address of the transfer program’s save area. The file handler must branch back to the transfer program by way of the return address in the transfer program’s Register 14.
Sequence of Function Calls for a User-Written File Handler

The following shows the sequence in which a transfer program calls the user-written file-handler functions during:

- A normal file transfer
- A file transfer with a transfer restart
- A file transfer with session recovery (LU 0 conversations only).

Normal File Transfer

Sending system

<table>
<thead>
<tr>
<th>User-written file handler</th>
<th>NetView FTP server</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN function /SM630000─► CALL</td>
<td></td>
</tr>
<tr>
<td>GET function /SM630000─► CALL</td>
<td></td>
</tr>
<tr>
<td>record, RC=ok─► send record</td>
<td></td>
</tr>
<tr>
<td>. .</td>
<td></td>
</tr>
<tr>
<td>. .</td>
<td></td>
</tr>
<tr>
<td>. .</td>
<td></td>
</tr>
<tr>
<td>. .</td>
<td></td>
</tr>
<tr>
<td>GET function /SM630000─► CALL</td>
<td></td>
</tr>
<tr>
<td>RC=end-of-file─► receive record</td>
<td></td>
</tr>
<tr>
<td>signal end-of-transfer</td>
<td></td>
</tr>
<tr>
<td>CLOSE function─► CALL</td>
<td></td>
</tr>
</tbody>
</table>

Receiving system

<table>
<thead>
<tr>
<th>NetView FTP server</th>
<th>User-written file handler</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL ────► OPEN function</td>
<td></td>
</tr>
<tr>
<td>receive record ────► PUT function</td>
<td></td>
</tr>
<tr>
<td>record ────► send record</td>
<td></td>
</tr>
<tr>
<td>. .</td>
<td></td>
</tr>
<tr>
<td>. .</td>
<td></td>
</tr>
<tr>
<td>. .</td>
<td></td>
</tr>
<tr>
<td>. .</td>
<td></td>
</tr>
<tr>
<td>CALL ────► CLOSE function</td>
<td></td>
</tr>
</tbody>
</table>

Figure 8. Sequence of Function Calls for a User-Written File Handler (Normal File Transfer)
Figure 9. Sequence of Function Calls for a User-Written File Handler. This sequence applies to a restarted file transfer.
File Transfer with Automatic Session Recovery (LU 0)

LU 6.2 does not support automatic session recovery for transfers between NetView FTP V2 MVS, NetView FTP VSE, and NetView FTP VM.

Sending System

<table>
<thead>
<tr>
<th>User-written file handler</th>
<th>FTP Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPEN function</td>
<td>CALL</td>
</tr>
<tr>
<td>GET function</td>
<td>CALL</td>
</tr>
<tr>
<td>record, RC=ok</td>
<td>send record</td>
</tr>
<tr>
<td>POINT function</td>
<td>CALL with</td>
</tr>
<tr>
<td>position depending on</td>
<td>restart info</td>
</tr>
<tr>
<td>restart info</td>
<td>from checkpoint</td>
</tr>
<tr>
<td>GET function</td>
<td>CALL</td>
</tr>
<tr>
<td>record, RC=ok</td>
<td>send record</td>
</tr>
<tr>
<td>GET function</td>
<td>CALL</td>
</tr>
<tr>
<td>RC=end-of-file</td>
<td>signal end-of-transfer</td>
</tr>
<tr>
<td>CLOSE function</td>
<td>CALL</td>
</tr>
</tbody>
</table>

Receiving System

<table>
<thead>
<tr>
<th>FTP Server</th>
<th>User-written file handler</th>
</tr>
</thead>
<tbody>
<tr>
<td>CALL</td>
<td>OPEN function</td>
</tr>
<tr>
<td>receive record</td>
<td>PUT function</td>
</tr>
<tr>
<td>CALL with record</td>
<td></td>
</tr>
<tr>
<td>receive record</td>
<td>PUT function</td>
</tr>
<tr>
<td>CALL with record</td>
<td></td>
</tr>
<tr>
<td>CALL</td>
<td>CLOSE function</td>
</tr>
</tbody>
</table>

Figure 10. Sequence of Function Calls for a User-Written File Handler with Automatic Session Recovery
Migrating a User-Written File Handler to NetView FTP V2.2 MVS

For a user-written file handler, you have to consider the following when migrating to NetView FTP V2.2 MVS:

- There is no longer a session recovery. Therefore, the FUEPOINT field of the user file-handler parameter list cannot contain the value “R”.

- The value “U” in the FUEPOINT field of the user file-handler parameter list indicates a restart. This can be a manual or an automatic restart. The field FUERSTPT indicates whether restart from checkpoint or restart from beginning has been requested.

- Ensure that a user-written file handler can provide a checkpoint record key. Depending on the contents of the FUEKNFTP field, the user-written file handler should (FUEKNFTP contains “N”) or should not provide a checkpoint record key. If the field FUEKNFTP contains “Y” and user-written file handler provides a checkpoint record key, NetView FTP ignores this information.
Chapter 3. Writing a Pre-Queuing User-Exit Routine

A pre-queuing user-exit routine can check and modify the parameters for a queue handler command that a user issued through one of the user interfaces. The queue handler command can be:

- ADD
- MODIFY
- RESTART
- All kinds of query commands, such as QUERY, QRYALL
- All kinds of delete commands, such as DELETE, DELALL, FORCEDEL.

When the transfer program passes the queue handler command to the pre-queuing user-exit routine, it has already set some default values (for the parameters that were missing from the queue handler command) into the application program parameter list (APL).

You create the pre-queuing user-exit routine as a separate load module that must have the name DVGCIRQX for NetView FTP V2 MVS and the name DVGIFRQX for NetView FTP VM and NetView FTP VSE:

For MVS  The pre-queuing user exit is part of all user-interface routines that both load and call the pre-queuing user-exit routine. You must link the module to a library that is contained in the current LINKLIST. The pre-queuing user exit routine will not be loaded from a STEPLIB.

For VSE  The pre-queuing user exit is part of all user-interface routines that both load and call the pre-queuing user-exit routine. You must link the phase to a library that is included in the search sequence given by the LIBDEF system control statement. If you want to prevent users from having their own private exit routines you can load the phase into the SVA. Then any user-exit routines must be SVA eligible.

For VM  The pre-queuing user exit is part of the queue handler that loads and calls the pre-queuing user-exit routine. You must link the module to a library that is accessible to the queue handler.

Therefore the request originator cannot use any private pre-queuing exit routines.

When the transfer program calls the exit routine, it provides the address of the APL, which contains all parameters for the queue handler commands.

The transfer program requires a pre-queuing user-exit routine. After you have installed the transfer program, the created load libraries contain a default pre-queuing user-exit routine that you must use if you do not provide your own routine. However, the default routine does not perform any checking or modifications. If the transfer program cannot find a pre-queuing user-exit routine, it rejects the issued queue handler command.

A member in the sample library of the installation tape for your transfer program contains a sample pre-queuing user-exit routine. For NetView FTP V2 MVS the member name is DVGCXARQ, for NetView FTP VSE and NetView FTP VM the member name is DVGSRQX.
What the Pre-Queuing User-Exit Routine Does

You can use the pre-queuing user-exit routine to perform the following:

- Check the parameters of a queue handler command and then either accept or reject the command. In this way you can restrict the use of the parameters. Before passing control to the exit, the transfer program checks the valid value range of the parameters. You may not want to allow all possible values for some parameters. For example, you may want to allow only the transfer mode “APLTO” (for send to). If the user specifies the transfer mode “APLFROM” (for retrieve from), you reject the request.

- Modify the following request parameter values:
  - Request password, field APLPSWD
  - Server class, field APLCLASS
  - Request priority, field APLPRTY
  - Remote server group name or remote LU name, and, if applicable, remote network ID, fields APLNODE, APLRLUNM, and APLNETID
  - Wait time, field APLWAITT.

Although a pre-queuing user-exit routine can change these parameters, their values must be specified correctly in the request. Otherwise, they will not pass the validity checks of the user-interface routines.

Does Not Apply to NetView FTP VSE

If the pre-queuing user-exit routine prevents the use of the MODIFY queue handler command, the operator can still modify the following file-transfer request parameters:

- Server class
- Request priority
- Remote LU name.

Modifying a Request Password

For the queue handler commands ADD and MODIFY and for all query and delete queue handler commands, the exit routine can modify the request password, which is stored in the APLPSWD field of the APL:

- For an ADD queue handler command, this field contains the password that the request originator uses to protect the request against unauthorized deletion.

- For a MODIFY queue handler command, this field contains the password that the issuer of the command uses to get access to the specified requests.

- For any query or delete queue handler command, this field contains the password that the issuer of the command uses to get access to the specified requests.

Modifying a request password can be useful for installations that do not want to distribute the master password to single users.

---

7 NetView FTP V2.2 MVS only.
Modifying a Server Class
For the ADD and MODIFY queue handler commands, the pre-queuing user-exit routine can modify the server class, which is stored in the APLCLASS field of the APL. For a discussion on server classes in your transfer program, see the *Installation, Operation, and Administration* book for your transfer program.

Modifying a server class can be useful for the following reasons:

- To prevent a user from giving an arbitrary server class to a file-transfer request.
- To ensure that a file-transfer request of a specific type is processed by the proper server.
- To achieve a balanced work load on all the started servers.

Modifying a Request Priority
For the ADD and MODIFY queue handler commands, the pre-queuing user-exit routine can modify a request priority, which is stored in the APLPRTY field of the APL.

For more information about assigning classes and priorities to requests, refer to the *User’s Guide* and *Installation, Operation, and Administration* for your transfer program.

Modifying a request priority can be useful in preventing a user from giving an arbitrary request priority to a file-transfer request.

Modifying a Server Group Name or Remote LU Name
For the ADD queue handler command, the exit routine can modify either the server group name or the remote LU name and, if applicable, the remote network ID, which are stored in the APLNODE, APLRLUNM, and APLNETID fields of the APL.

Modifying a server group name or remote LU name can be useful for the following reasons:

- To ensure that a file-transfer request of a specific type is processed by a remote server that performs requests of this type.
- To achieve a balanced work load on all the started servers of a remote node.
Modifying the Wait Time

Applies to NetView FTP V2 MVS Only

For the ADD and the QUERY queue handler commands, the exit routine can modify the time interval for which the batch job interface routine waits until the next QUERY queue handler command is issued. The wait time is stored in the APLWAITT field of the APL.

Modifying the wait time can be useful for the following reasons:

- To prevent the need of a program fix to change the wait time
- To set a default value dependent on the customer installation
- To reduce the usage of resources when the specified wait time was too small.

Changing Parameters

The fields the pre-queuing user-exit routine can change are summarized in the following table. Modifications to these fields are accepted only if an appropriate return code is set. The following conventions are used:

<table>
<thead>
<tr>
<th>Field Name</th>
<th>NetView FTP V2 MVS</th>
<th>NetView FTP VM</th>
<th>NetView FTP VSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>APLCLASS</td>
<td>A+M+Q</td>
<td>A+M</td>
<td>A+M</td>
</tr>
<tr>
<td>APLRTY</td>
<td>A+M</td>
<td>A+M</td>
<td>A+M</td>
</tr>
<tr>
<td>APLPSWD</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>APLRLUNM</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>APLNETID</td>
<td>A</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>APLNODE</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>APLWAITT</td>
<td>A+Q</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Note: When you modify any field with an ADD function, the modification is reflected in the request. In a MODIFY function, only the modification of APLCLASS and APLRTY fields are reflected in the request. All other functions use the modification as a search argument.
Coding the Pre-Queuing User-Exit Routine

The following gives guidelines for coding the pre-queuing user-exit routine. If you code the pre-queuing user-exit routine in a programming language other than Assembler, PL/I, REXX, or C language, you must define the structure of the APL.

Using the Application Program Parameter List (APL)

The APL contains all parameters that the request originator specifies. For a detailed description of the APL, see Chapter 12, “Using the APL” on page 111.

Note: NetView FTP passes only the information needed by the partner transfer system. As a result, not all APL fields contain values that user-exit routines can use.

Using the Application Program Parameter List Extension (APX)

Applies to NetView FTP V2 MVS Only

When a PDS is transferred, the APX contains a header and an entry for each member of the PDS to be transferred, up to the specified size of the APX. For a detailed description of the APX, see Chapter 14, “Using the APX” on page 143.

End of Applies to NetView FTP V2 MVS Only

Using the Registers

The register conventions between the transfer program and the user-written exit routines follow normal linkage conventions.

On entry to the exit routine, Register 1 points to a fullword that contains the address of the APL. The following overview lists the contents of all general registers on entry to the exit routine:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>1</td>
<td>Points to the address of the APL</td>
</tr>
<tr>
<td>2 to 12</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>13</td>
<td>Contains the address of a save area of 18 fullwords in the transfer program</td>
</tr>
<tr>
<td>14</td>
<td>Contains the return address to the transfer program</td>
</tr>
<tr>
<td>15</td>
<td>Contains the entry address to the pre-queuing routine</td>
</tr>
</tbody>
</table>

---

8 PL/I does not apply to NetView FTP VSE.
9 REXX and C language apply to NetView FTP V2 MVS only.
**Saving the Registers**
On entry to a pre-queuing exit routine, the routine must store Registers 0 to 12, 14 and 15 into the save area to which Register 13 points. The routine must use fullwords 4 to 18 for this purpose.

The pre-queuing exit routine must also have a save area of 18 fullwords. On entry to a user-written routine, the routine must store Register 13 (the transfer program’s save area address) in the second word of the exit routine’s save area. The third word of the transfer program’s save area must contain the address of the exit routine’s save area.

After saving the transfer program’s registers, a user-written exit routine can use the registers according to its own needs.

**Returning to the Transfer Program**
Before the routine returns control to the transfer program, it must reload the transfer program’s registers from the transfer program’s save area. The routine must reset Register 13 to the address of the transfer program’s save area. The exit routine must branch back to the transfer program by way of the return address in the transfer program’s Register 14.

**Specifying Return Codes**
Based on the results of its processing, the exit routine must insert a return code into the APLRC field of the APL. The transfer program uses this code to set up a return code and a reason code for the file-transfer request.

The following return codes are valid for the APLRC field:

0  The pre-queuing user-exit routine accepts the file-transfer request without modifying it.

2  The pre-queuing user-exit routine accepts the file-transfer request, but modifies one or more fields in the APL.

   If the user-exit routine modifies the request, the return code must be set to 2. Otherwise the modifications are not accepted.

> 2  The pre-queuing user-exit routine rejects the file-transfer request. If, in this case, the code is between 3 and 999, the transfer program adds 1000 to it and uses the result as the reason code for the file-transfer request. You can, therefore, make the reason for the rejection available to the request originator.
Chapter 4. Writing a Pre-Transfer User-Exit Routine

The optional pre-transfer user-exit routine enables you to perform any processing including checking and changing a file transfer request, before a server starts the actual transfer of data.

Normally, during a file transfer, the transfer program calls the pre-transfer user-exit routine at the beginning of a transfer, the post-transfer user-exit routine at the end of a transfer, and the post-conversation\(^{10}\) user-exit routine when the conversation is terminated. The pre-transfer user-exit routine at the sending system is called first, then the transfer program sends the (possibly modified) data that is relevant to the file transfer to the receiving system where the receiving transfer program presents it to its pre-transfer user-exit routine.

You create the pre-transfer user-exit routine as a separate load module. The routine’s name must not be FTPSECUR, which is reserved for the FTP V2 security/statistics collection user-exit routine. Refer to Chapter 9, “Writing a Security/Statistics Collection User-Exit Routine” on page 101 for more information about the security/statistics collection user-exit routine.

A server loads the pre-transfer user-exit routine into its storage area during its initialization processing. The name of the exit routine can be specified by the PRETRAN\(^{11}\) or PPEXIT server initialization parameter, which is described in Installation, Operation, and Administration. Different servers can have different pre-transfer user-exit routines.

--- Does Not Apply to NetView FTP V2 MVS ---

You must have both a pre-transfer user-exit routine and a post-transfer user-exit routine. However, you can have both routines integrated into one module. How to do this is described in Chapter 8, “Considerations when Designing User-Exit Routines” on page 93.

--- End of Does Not Apply to NetView FTP V2 MVS ---

\(^{10}\) The post-conversation user-exit routine applies to NetView FTP V2 MVS only.

\(^{11}\) PRETRAN applies to NetView FTP V2 MVS only.
You can have one of the following:

- A pre-transfer user-exit routine
- A pre-transfer user-exit routine and a post-transfer user-exit routine
- A pre-transfer user-exit routine, a post-transfer user-exit routine, and a post-conversation user-exit routine.

In this case, all routines can be integrated into one module. How to do this is described in Chapter 8, “Considerations when Designing User-Exit Routines” on page 93.

A pre-transfer user-exit routine can communicate with a post-transfer and a post-conversation user-exit routine via local and global data areas, see Chapter 7, “Communicating with User-Exit Routines” on page 83 for more information.

**Note:** For MVS, because the server must be linked to an APF authorized library, you must also link your user-exit routine to an APF authorized library. For VSE, the user-exit routine must be cataloged in one of the libraries specified in the LIBDEF system control statements of the partition startup job.

A member in the sample library of the installation tape for your transfer program contains a sample pre-transfer user-exit routine. The member is called DVGCXAPT for NetView FTP V2 MVS, for NetView FTP VM, and NetView FTP VSE.
What the Pre-Transfer User-Exit Routine Does

When the transfer program calls the pre-transfer user-exit routine, it inserts the parameters of the file-transfer request into the APL and passes it to the exit routine.

After a pre-transfer user-exit routine gets control, it can perform any processing, including one or all of the following:

- Parameter checking for the impending file transfer:
  Depending on the return code from the pre-transfer user-exit routine, the file transfer can be stopped before a server begins the actual transfer of data. The rejection can be temporary or permanent, depending on the return code that the exit routine passes to the transfer program. If the exit routine requests a permanent rejection, the transfer program treats the file-transfer request as finished, and does not schedule it again.
  If the exit routine requests a temporary rejection, the transfer program treats the file-transfer request as waiting or finished, depending on the value of the Automatic Transfer Restart parameter.

- Preparation activities for the impending file transfer, such as the invocation of a utility program to prepare the file to be transferred.

- Changing the parameter values in the file-transfer request that apply to the node on which the routine runs.

Applies to NetView FTP V2 MVS Only

- Setting an indicator for the transfer program whether the security environment has been verified by the pre-transfer user-exit routine. If so, NetView FTP does not verify the password.

End of Applies to NetView FTP V2 MVS Only
Changing Parameters

You can tell the pre-transfer user-exit routine to change the values of the APL file-transfer request parameters shown in the following table. Modifications to these fields are accepted only if an appropriate return code is set.

The following conventions are used:

**S** Fields can be changed at the sending system. If the user-exit routine at the requesting system makes a change, the changed parameter value is passed to the user-exit routine at the responding system.

**R** Fields can be changed at the receiving system.

**A** Parameter value is passed to the user-exit routine at the responding system.

— Does not apply.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>NetView FTP V2.2</th>
<th>NetView FTP VM</th>
<th>NetView FTP VSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>APLAPXPT</td>
<td>S+R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLCOMPR</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLCONVT</td>
<td>S+R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLDYDE</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLDYIE</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLDYKLN</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLDYKO</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLDYME</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLDYMPW</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLDYRSA</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLDYUCP</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APDGXDLN</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLGDPT</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APPLXDLN</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLLXDPT</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLPPXIN</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLRACCO</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRACGI</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRACPW</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRACU2</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRAVBL</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRCCSI</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRDIRB</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRDSS</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDVPW</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYBL</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYDD</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYDN</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYEX</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYFD</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYKF</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYLI</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYLR</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYLS</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRDYMD</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYMM</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>Field Name</td>
<td>NetView FTP V2.2 MVS</td>
<td>NetView FTP VM</td>
<td>NetView FTP VSE</td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>APLRDYPH</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYQU</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYRE</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYTD</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRDYUC</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRDYUT</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRENCR</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLREOPS</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLREOPU</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRFLAB</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRFORG</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRFFTYP</td>
<td>-</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRJNO1</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJNO2</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJNO3</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJNO4</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJNO5</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJNO6</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJOK1</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJOK2</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJOK3</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJOK4</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJOK5</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJOK6</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRJPDS</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRLFA</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRLFL</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRNID</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRNUID</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLROWID</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRPADD</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRPOID</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRPROC</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSAVG</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSDAT</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSLIK</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSMGM</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSOPT</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSPRM</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRSREF</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSSEC</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRSSEG</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSSEP</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSSTO</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSTPT</td>
<td>R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLRSUD1</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRSUD2</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRSUD3</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRSUD4</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRSUD5</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRSUD6</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRSUD7</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRSUD8</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
</tbody>
</table>

Chapter 4. Writing a Pre-Transfer User-Exit Routine
<table>
<thead>
<tr>
<th>Field Name</th>
<th>NetView FTP V2.2 MVS</th>
<th>NetView FTP VM</th>
<th>NetView FTP VSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>APLRSUNI</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRVVID</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRVEX3</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRVEXT</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRVFML</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRVIN3</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRVINT</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLRVOC</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLRPWP</td>
<td>-</td>
<td>R</td>
<td>-</td>
</tr>
<tr>
<td>APLSACCO</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSACGI</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSACPW</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSACUJ</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSACU2</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSCCSI</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSDSS</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSDVPW</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSDYBL</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSDYDD</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSDYFD</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSDYL1</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSDYL2</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSDYMM</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSDYPH</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSDYQU</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSECV</td>
<td>S+R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLENCR</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLEOPS</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLEOR</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSFLAB</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSFORG</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSFTYP</td>
<td>-</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSFHMD</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJNO1</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJNO2</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJNO3</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJNO4</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJNO5</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJNO6</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJOK1</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJOK2</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJOK3</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJOK4</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJOK5</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJOK6</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSJPDS</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSLFA</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSLFL</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APLSNMID</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSNU1D</td>
<td>S</td>
<td>S</td>
<td>S</td>
</tr>
<tr>
<td>APLSOWID</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>Field Name</td>
<td>NetView FTP V2.2 MVS</td>
<td>NetView FTP VM</td>
<td>NetView FTP VSE</td>
</tr>
<tr>
<td>------------</td>
<td>----------------------</td>
<td>----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>APLSPOID</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSSUD1</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSSUD2</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSSUD3</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSSUD4</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSSUD5</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSSUD6</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSSUD7</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSSUD8</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSVVDID</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSVEX3</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSVEXT</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSVFML</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSVIN3</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSVINT</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLSVPW</td>
<td>-</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>APLUEUI0</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUEUI1</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUEUI2</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUEUI3</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUEUI4</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUEUI5</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUEUI6</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUEUI7</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUEUI8</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUEUI9</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUXLEN</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLUXPTR</td>
<td>S+R</td>
<td>S+R</td>
<td>S+R</td>
</tr>
<tr>
<td>APLVTYP</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td>APLXOPT</td>
<td>S</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APXENT</td>
<td>S+R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APXLEN</td>
<td>S+R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APXOPT</td>
<td>S+R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APXRNME</td>
<td>S+R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APXSNAME</td>
<td>S+R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APXTYP</td>
<td>S+R</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>APXXNME</td>
<td>S+R</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

You can find a discussion of these fields in Chapter 12, "Using the APL" on page 111, and in the User’s Guide for NetView FTP VM and NetView FTP VSE and in the NetView FTP Parameter Reference for NetView FTP V2 MVS.
C au ng a Pre-Transfer User-Exit Routine

The following gives guidelines for coding a pre-transfer user-exit routine.

Calling the Pre-Transfer User-Exit Routine

If specified in the server startup procedure, the transfer program calls a pre-transfer user-exit routine immediately before it attempts to transfer a file. Up to this point, the transfer program has not allocated or opened the file it is to send or to receive. Also, the transfer program has not yet built the temporary security environment.

Normally, during a file transfer, the transfer program calls the pre-transfer user-exit routine at the beginning and the post-transfer user-exit routine at the end of a transfer. The pre-transfer user-exit routine in the sending system is called first, then the transfer program sends data that is relevant to the file transfer to the receiving system where the receiving transfer program presents it to its pre-transfer user-exit routine.

However, if an error occurs before or while the sending transfer program is transferring this data to the receiving transfer program, the receiving transfer program calls only the post-transfer user-exit routine. The same can happen in the sending system, when the data relevant to the file-transfer is obtained by the receiving transfer program from its request queue, but the receiving transfer program fails to transfer this data to the sending system. In these cases the transfer program indicates, by setting the APLXMODE field to blank, to the post-transfer user-exit routine that no file-transfer request is available. See Chapter 8, “Considerations when Designing User-Exit Routines” on page 93 for an overview of the sequences in which the transfer program calls the user-exit routines.

The name of the pre-transfer user-exit routine you want to call is specified by the server initialization parameter. The server initialization parameters are described in the Installation, Operation, and Administration book for your transfer program.

The transfer program indicates the function for which it calls the module by the APLVBC field of the APL. Setting the value of APLVBC to APLPRERE (‘P’) indicates that the transfer program requires the module to work as a pre-transfer user-exit routine.

Does Not Apply to NetView FTP VSE

Although a server can conduct more than one file transfer at a time, the exit routine does not have to be reentrant. However, coding exit routines as reentrant ensures that you do not have multiple copies of the same routine in a server’s storage area.

End of Does Not Apply to NetView FTP VSE

An exit routine must, in all cases, be serially reusable. The sequence in which the transfer program calls an exit routine for the various parallel file transfers is unpredictable.
Using the Application Program Parameter List (APL)

The APL contains most or all of the parameters of a file-transfer request. See Chapter 12, “Using the APL” on page 111 for a detailed description of the APL.

Using the Application Program Parameter List Extension (APX)

| Applies to NetView FTP V2 MVS Only |

When a list of PDS members is transferred, an APX is provided for the pre-transfer user-exit routine. The address of the APX is stored in the APLAPXPT field of the APL. The APX contains a header and an entry for each member of the PDS specified in the file-transfer request. So, the APX is of variable length. If your exit routine makes any changes to the APX, you must ensure that the APXENT and APXLEN fields of the APX are updated to reflect the changes. If the APX is not large enough, you must increase its size. For example, you can use the GETMAIN macro (or similar macro) to do this. Do not try to free an existing area for the APX.

Passing User-Specific Information

You can use the pre-transfer user-exit routine to pass user-specific information to the next user-exit routine to be called. For example, you can request your users to specify security information that is then checked by your user-exit routine.

The user-specific information is specified by the Input for a User-Exit Routine parameter. The size (in bytes) of this user-specific information is given in Chapter 7, “Communicating with User-Exit Routines” on page 83, together with the transfer programs to which they apply.

Using the Registers

The register conventions between the transfer program and the user-written exit routine follow normal linkage conventions.

On entry to the exit routine, Register 1 points to a fullword that contains the address of the APL. On entry to the user-written exit routine, the general registers have the following contents:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>1</td>
<td>Points to the address of the APL</td>
</tr>
<tr>
<td>2 to 12</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>13</td>
<td>Contains the address of a save area of 18 fullwords in the transfer program</td>
</tr>
<tr>
<td>14</td>
<td>Contains the return address to the transfer program</td>
</tr>
<tr>
<td>15</td>
<td>Contains the entry address to the exit routine.</td>
</tr>
</tbody>
</table>
Saving the Registers
On entry to an exit routine, the routine must store Registers 0 to 12, 14 and 15 into the save area to which Register 13 points. The routine must use fullwords 4 to 18 for this purpose.

The exit routine must also have a save area of 18 fullwords. On entry to a user-written routine, the routine must store Register 13 (the transfer program’s save area address) in the second word of the exit routine’s save area. The third word of the transfer program’s save area must contain the address of the exit routine’s save area.

After saving the transfer program’s registers, a user-written exit routine can use the registers according to its own needs.

Returning to the Transfer Program
Before the routine returns control to the transfer program, it must reload the transfer program’s registers from the transfer program’s save area. The routine must reset Register 13 to the address of the transfer program’s save area. The exit routine must branch back to the transfer program by way of the return address in the transfer program’s Register 14.

Specifying Return Codes
A pre-transfer user-exit routine must indicate the result of its processing by inserting a return code into the APLRC field of the APL. This code controls the transfer program’s processing of the file-transfer request.

The following return codes are valid for the APLRC field:

0 The pre-transfer user-exit routine accepts the file-transfer request without modifying it. When the transfer program calls a pre-transfer user-exit routine, it initializes APLRC to this value.

2 The pre-transfer user-exit routine accepts the file-transfer request, but modifies one or more fields in the APL.

If the user-exit routine modifies the request, the return code must be set to 2. Otherwise the modifications are not accepted.

4 The pre-transfer user-exit routine rejects the file-transfer request temporarily (for example, because of a temporary allocation failure). The transfer program sets up a return code and a reason code for the file transfer and writes the file-transfer request back to the request queue. The status of the file-transfer request depends on the value of the automatic transfer restart parameter.

> 4 The user-exit routine rejects the file-transfer request. The transfer program sets up a return code and a reason code for the file transfer and writes the file-transfer request back to the request queue after it has given it the status finished.
Passing Messages to the Transfer Program
The pre-transfer user-exit routine can pass messages to the transfer program to have these messages printed as normal transfer program messages. The transfer program routes these messages to the transfer report file and the server log file (report collection file for VSE). See Chapter 10, “Issuing Messages from User-Exit Routines” on page 103, for more information.

Using the Trace Facility
The pre-transfer exit routine can make use of the trace facility of the transfer program. The transfer program can trace both the interface between a server and the exit routine, and an area defined by the exit routine. See Chapter 11, “Using the Trace Facility” on page 107, for more information.
Chapter 5. Writing a Post-Transfer User-Exit Routine

With the optional post-transfer user-exit routine you can perform any processing, including looking at a file-transfer request’s parameters and collecting information after the transfer program completes the transfer.

You create the post-transfer user-exit routine as a separate load module. The routine’s name cannot be FTPSECUR, which is reserved for the FTP V2 security/statistics collection user-exit routine.

A server loads the post-transfer user-exit routine into its storage area during its initialization processing. The name of the exit routine can be specified by the PPEXIT, or POSTTRAN\(^{12}\) server initialization parameter. If you run multiple servers in parallel, you can have a different post-transfer user-exit routine for each server.

Does Not Apply to NetView FTP V2 MVS

You cannot have just a post-transfer user-exit routine. You must have both a pre-transfer user-exit routine and a post-transfer user-exit routine. However, you can have both routines integrated into one module. How to do this is described in Chapter 8, “Considerations when Designing User-Exit Routines” on page 93.

End of Does Not Apply to NetView FTP V2 MVS

Applies to NetView FTP V2 MVS Only

You can have one of the following:

- A pre-transfer user-exit routine
- A post-transfer user-exit routine
- A pre-transfer user-exit routine and a post-transfer user-exit routine
- A pre-transfer user-exit routine, a post-transfer user-exit routine, and a post-conversation user-exit routine.

In this case, all routines can be integrated into one module. How to do this is described in Chapter 8, “Considerations when Designing User-Exit Routines” on page 93.

A post-transfer user-exit routine can communicate with a pre-transfer and a post-conversation user-exit routine via local and global data areas, see Chapter 7, “Communicating with User-Exit Routines” on page 83 for more information.

End of Applies to NetView FTP V2 MVS Only

Note: For MVS, because the server must be linked to an APF authorized library, you must also link the user-exit routine to an APF authorized library. For VSE, the user-exit routine must be cataloged in one of the libraries specified in the LIBDEF system control statements.

\(^{12}\) POSTTRAN applies to NetView FTP V2 MVS only.

© Copyright IBM Corp. 1988, 1994

55
A member in the sample library of the installation tape for your transfer program contains a sample post-transfer user-exit routine. The member is called DVGCXACO.

What the Post-Transfer User-Exit Routine Does

When the transfer program calls the post-transfer user-exit routine, it inserts the parameter values of the file-transfer request into the APL, adds information about the file transfer to it, and passes it to the user-exit routine. You can use the post-transfer user-exit routine to perform the following:

- Collect statistical information about a previous file transfer.
- Post-processing activities. For example, you can invoke a utility to process the receiving file.

Changing Parameters

You can tell the post-transfer user-exit routine to change the values of the APL file-transfer parameters shown in the following table. Modifications to these fields are accepted only if an appropriate return code is set.

The following conventions are used:

- **S** Fields can be changed at the sending system.
- **R** Fields can be changed at the receiving system.
- **—** Does not apply.

<table>
<thead>
<tr>
<th>Field Name</th>
<th>NetView FTP V2</th>
<th>MVS</th>
<th>NetView FTP VM</th>
<th>S+R</th>
<th>NetView FTP VSE</th>
<th>S+R</th>
</tr>
</thead>
<tbody>
<tr>
<td>APLGXDLN</td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
</tr>
<tr>
<td>APLGXDPT</td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
</tr>
<tr>
<td>APLLXDLN</td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
</tr>
<tr>
<td>APLLXDPT</td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
</tr>
<tr>
<td>APLPPXIN</td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
</tr>
<tr>
<td>APLUXLEN</td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
</tr>
<tr>
<td>APLUXPTR</td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
<td>S+R</td>
<td></td>
</tr>
</tbody>
</table>

Coding a Post-Transfer User-Exit Routine

The following gives guidelines for coding a post-transfer user-exit routine.

Calling a Post-Transfer User-Exit Routine

A post-transfer user-exit routine gets control, at both the sending and receiving systems, after the file transfer has finished, either successfully or unsuccessfully.

If NetView FTP has already allocated a file, the file remains allocated on entry to the user-exit routine, but it is closed. The temporary security environment still exists if the transfer program has created one at the beginning of the file transfer. See Chapter 8, “Considerations when Designing User-Exit Routines” on page 93, for an overview of the sequences in which the transfer program calls the pre-transfer and post-transfer user-exit routines.
From this overview you can see that a post-transfer user-exit routine can be called by the transfer program without a pre-transfer user-exit routine having been called previously. In this case, it is possible that the transfer program did not create a temporary security environment and did not dynamically allocate the sending or receiving file.

If you specify only one name in the PPEXIT server initialization parameter, one load module serves both as a pre-transfer user-exit and as a post-transfer user-exit routine. For MVS only, you can also specify the POSTTRAN server initialization parameter to call a post-transfer user-exit routine. For more information about the PPEXIT and POSTTRAN server initialization parameters, refer to the Installation, Operation, and Administration book for your transfer program. The transfer program indicates the function for which it calls the module by the verb code field, APLVBC, of the APL. Setting the value of APLVBC to APLCOMRE (C) indicates that the transfer program requires the module to work as a post-transfer user-exit routine.

Although a server can conduct more than one file transfer at a time, the exit routine does not have to be reentrant. However, coding exit routines as reentrant ensures that you do not have several copies of the same routine in a server's storage area. An exit routine must, in all cases, be serially reusable. The sequence in which the transfer program calls an exit routine for the various parallel file transfers is unpredictable.

Using the Application Program Parameter List (APL)

The APL contains the file-transfer request's parameter values. Additionally, it contains statistical information that the transfer program has prepared at the end of the transfer. This information is described under “Special Fields for Pre-Transfer, Post-Transfer, and (for MVS) Post-Conversation User-Exit Routines” on page 135.

The transfer program provides a return code, in the APLRC field of the APL, from which the post-transfer user-exit routine can determine whether the transfer was successful or not. See the symbolic literals APLOK, APLERROR, and APLABEND in “Feedback Fields and Verb Code” on page 113.
Passing User-Specific Information

You can use the post-transfer user-exit routine to pass user-specific information to the next user-exit routine to be called. For example, you can request your users to specify security information which is then checked by your user-exit routine.

Information from the pre-transfer user-exit routine at the local or remote system can be passed to the post-transfer user-exit routine at the local or remote system. For more information, see Chapter 7, “Communicating with User-Exit Routines” on page 83.

Using the Registers

The register conventions between the transfer program and the user-written exit routine follow normal linkage conventions.

On entry to the exit routine, Register 1 points to a fullword that contains the addresses of the APL. On entry to the user-written exit routine, the general registers have the following contents:

<table>
<thead>
<tr>
<th>Register</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>1</td>
<td>Points to the address of the APL</td>
</tr>
<tr>
<td>2 to 12</td>
<td>Unpredictable</td>
</tr>
<tr>
<td>13</td>
<td>Contains the address of a save area of 18 fullwords in the transfer program</td>
</tr>
<tr>
<td>14</td>
<td>Contains the return address to the transfer program</td>
</tr>
<tr>
<td>15</td>
<td>Contains the entry address to the exit routine</td>
</tr>
</tbody>
</table>

Saving the Registers

On entry to an exit routine, the routine must store Registers 0 to 12, 14 and 15 into the save area to which Register 13 points. The routine must use fullwords 4 to 18 for this purpose.

The exit routine must also have a save area of 18 fullwords. On entry to a user-written routine, the routine must store Register 13 (the transfer program’s save area address) in the second word of the exit routine’s save area. The third word of the transfer program’s save area must contain the address of the exit routine’s save area.

After saving the transfer program’s registers completely, a user-written exit routine can use the registers according to its own needs.

Returning to the Transfer Program

Before the routine returns control to NetView FTP, it must reload the transfer program’s registers from the transfer program’s save area. The routine must reset Register 13 to the address of the transfer program’s save area. The exit routine must branch back to the transfer program by way of the return address in the transfer program’s Register 14.
Specifying Return Codes

The transfer program does not process any return code of a post-transfer user-exit routine.

Passing Messages to the Transfer Program

The post-transfer user-exit routine can pass messages to the transfer program to have these messages printed as normal transfer program messages. The transfer program routes these messages to the transfer report file and the server log file. See Chapter 10, "Issuing Messages from User-Exit Routines" on page 103, for more information.

Using the Trace Facility

The post-transfer user-exit routine can use the trace facility of the transfer program. The transfer program can trace both the interface between a server and the exit routine, and an area defined by the exit routine. See Chapter 11, "Using the Trace Facility" on page 107, for more information.

Migrating DVGJOBS to NetView FTP V2.2 MVS

To continue to use an existing version of DVGJOBS when you migrate to NetView FTP V2.2 MVS you need only recompile DVGJOBS with the new APL. However, for transfers between two NetView FTP V2 MVS systems, you can use the additional features of the post-transfer jobs function. The post-transfer jobs function is described in detail in the NetView FTP Parameter Reference and in the NetView FTP V2 MVS User’s Guide.

NetView FTP V2 MVS provides a program DVGCXAPR in the samples library, which can be used to set a job step return code in the post-transfer job according to the return code, reason code, or status of the NetView FTP file transfer. Subsequent steps can then be executed under control of the job step return code.

Symbolic parameters can be used in the post-transfer job to represent the file transfer return code (&DVGRETCD), reason code (&DVGRSNCD), and the file transfer status (&DVGTRST).

You invoke DVGCXAPR in the following way:

```
//STEPRC EXEC PGM=DVGCXAPR,PARM='&DVGRSTD'
```

DVGCXAPR does the following:

1. Returns code zero when no parameter is provided.
2. Checks that parameter and returns code 9999 when the parameter is longer than 4 bytes, or if the parameter contains nonnumerics and contains more than one byte.
3. Converts the numeric parameter to fixed binary, and loads the character parameter as a return code. For example, 198 for character F, or 230 for character W.
4. Returns the parameter value in register 15 to the caller.
The following is an example of a job that calls DVGCXAPR:

```plaintext
//uid JOB (accn,n),uid,MSGLEVEL=(1,1),MSGCLASS=X,
//   CLASS=D,NOTIFY=uid,USER=uid
//******************************************************************************
// * translate NFTP return code to step return code *
//******************************************************************************
//  FTPREC EXEC PGM=DVGCXAPR,PARM=&DVGRETCD
//SYSPRINT DO SYSOUT**
// * INPUT: FTP return code
// * OUTPUT: Step return code
//******************************************************************************
// * translate NFTP reason code to step return code *
//******************************************************************************
//  FTPRSN EXEC PGM=DVGCXAPR,PARM=&DVGRSNCD
//SYSPRINT DO SYSOUT**
// * INPUT: FTP reason code
// * OUTPUT: Step return code
//******************************************************************************
// * translate NFTP queuing status to step return code *
//******************************************************************************
//  FTPQST EXEC PGM=DVGCXAPR,PARM=&DVGTRST
//SYSPRINT DO SYSOUT**
// * INPUT: FTP reason code
// * OUTPUT: Step return code
//******************************************************************************
// Transfer status finished 'F' generates return code : 198
// Transfer status deleted 'D' generates return code : 196
// Transfer status on hold 'H' generates return code : 200
// Transfer status waiting 'W' generates return code : 230
// Transfer status active 'A' generates return code : 193
//******************************************************************************
// now start processing depending on NFTP return or reason code
//******************************************************************************
//user1 EXEC PGM=userprog,COND=((0,GE,FTPRETC),(8,LT,FTPRETC))
//SYSPRINT DO SYSOUT**
//******************************************************************************
//user2 EXEC PGM=userprog,COND=((0,GE,FTPRETC),(nnn,EQ,FTPRSN))
//SYSPRINT DO SYSOUT**
```
Examples of Using DVGJOBS in NetView FTP MVS

Assume there is a job library NFTP.JOBS with jobs JOBG and JOBB at both the sending and the receiving system:

1. `PPXINFO='CS=(1,NFTP.JOBS,JOBG)'`
   
   Submit job JOBG from NFTP.JOBS at the sending system if the file transfer is successful.

2. `PPXINFO='CR=(1,NFTP.JOBS,,JOBB)'`
   
   Submit job JOBB from NFTP.JOBS at the receiving system if the file transfer was unsuccessful.

3. `PPXINFO='CS=(1,NFTP.JOBS,JOBG,JOBB)'`
   
   Submit job JOBG from NFTP.JOBS at the sending system if the file transfer was successful. Submit job JOBB from NFTP.JOBS at the sending system if the file transfer was unsuccessful.

4. `PPXINFO='CR=(1,,,JOBB,28)'`
   
   Submit job JOBB from DVG.JOBLIB at the receiving system if the file transfer was unsuccessful and ended with reason code 28.

5. `PPXINFO='CS=(1,,,JOBB,F)'`
   
   Submit job JOBB from DVG.JOBLIB at the sending system if the file transfer was unsuccessful and if the request was given the status F (for finished).

6. `PPXINFO='CR=(1,,,JOBB,F) CR=(1,NFTP.JOBS,,JOBG,W)'`
   
   Submit job JOBB from DVG.JOBLIB at the receiving system if the file transfer was unsuccessful and if the request was given the status F (for finished). Submit JOBG from NFTP.JOBS at the receiving system if the file transfer was unsuccessful and the request was given the status W (for waiting).

How DVGJOBS Uses the Value of the Exit Routine Input Parameter

When the transfer program passes control to DVGJOBS, DVGJOBS scans the value of the User-Exit Routine Input parameter and determines what it should do:

- If the parameter is not specified, DVGJOBS passes control back to the transfer program, without performing any functions.
- If the parameter is specified and is syntactically correct, DVGJOBS attempts to carry out the specified function. If it is unable to carry out the function, DVGJOBS writes an error message to the transfer program.
- If the parameter is specified but is syntactically incorrect, DVGJOBS writes an error message to the transfer program.
How DVGJOBS Modifies Submitted Jobs

Before submitting a job, DVGJOBS can override the values of some of the job’s JCL parameters. For example, if the job processes the receiving data set, DVGJOBS can insert the name of the receiving data set into the job.

If you specify the variables that DVGJOBS can replace in the JCL code for a job that is to be submitted, DVGJOBS replaces the variable with the corresponding value from the file-transfer request before submitting the job to the job entry subsystem. NetView FTP passes each replacement parameter value to DVGJOBS via one of two fields of the DVGAPL control block, depending on whether the job is submitted at the sending or the receiving system. Figure 11 on page 64 contains a list of all variables that DVGJOBS can replace, together with a description of the associated parameter and the names of the DVGAPL fields that the transfer program uses to pass the corresponding values for the sending and receiving nodes.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Parameter Description</th>
<th>DVGAPL Field for:</th>
</tr>
</thead>
<tbody>
<tr>
<td>DVGACUI</td>
<td>Security system user ID of request originator (up to 8 chars)</td>
<td>Sending System</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Receiving System</td>
</tr>
<tr>
<td>DVGACPW</td>
<td>Security system user password of request originator</td>
<td>APLSACPW</td>
</tr>
<tr>
<td></td>
<td></td>
<td>APLRACPW</td>
</tr>
<tr>
<td>DVGACGI</td>
<td>Security system group ID of request originator</td>
<td>APLSACGI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>APLRACGI</td>
</tr>
<tr>
<td>DVGDYFD</td>
<td>Name of data set to be transferred</td>
<td>APLSDYFD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>APLRDYFD</td>
</tr>
<tr>
<td>DVGDYMM</td>
<td>Name of a PDS member to be transferred</td>
<td>APLSDYMM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>APLRDYMM</td>
</tr>
<tr>
<td>DVGDYLI</td>
<td>First volume serial number</td>
<td>APLSDYLI</td>
</tr>
<tr>
<td></td>
<td></td>
<td>APLRDYLI</td>
</tr>
</tbody>
</table>

The rules for specifying the variables in the JCL statements are the same as the MVS JCL rules.

Variables can be specified anywhere in columns 1 to 71, except in comment lines (lines with the characters //* in columns 1 to 3).

As DVGJOBS does not split records, the user must ensure that there is enough space in each input record to hold the replacement value. It is good practice to have a complete JCL record for each variable.
Example of How DVGJOBS Replaces Variables in a JCL Input Member

Assume that DVGJOBS is invoked at the receiving system and that the following DVGAPL fields contain the following values:

- **APLRACUI**: 'USER1'
- **APLRACPW**: 'USERPW1'
- **APLRACGI**: 'GROUP1'
- **APLRDYFD**: 'FTP.RECEIVED.PDSABCD'
- **APLRDYMM**: 'MEMBERXY'

Submitting the job shown in Figure 12 has the same effect as submitting the job shown in Figure 13.

The variables can be specified in the JCL statements as shown in Figure 12 on page 65, or in the batch-job control statements as shown in Figure 14.

```
//&DVGACUI.A JOB (1234,5),USER=&DVGACUI,PASSWORD=&DVGACPW,
// GROUP=&DVGACGI,NOTIFY=&DVGACUI,MSGCLASS=A,CLASS=A
//STEP EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=&DVGDYFD.(&DVGDYMM),
// DISP=SHR
//SYSUT2 DD DSN=&DVGDACUI..&DVGDYFD..&DVGDYMM,
// DCB=(DSORG=PS,RECFM=FB,BLKSIZE=3120),
// UNIT=SYSDA,SPACE=(TRK,(5,5),RLSE),DISP=(NEW,CATLG,DELETE)
//SYSIN DD DUMMY
```

**Figure 12. Sample JCL Input Member**

```
//USER1A JOB (1234,5),USER=USER1,PASSWORD=USERPW1,
// GROUP=GROUP1,NOTIFY=USER1,MSGCLASS=A,CLASS=A
//STEP EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=A
//SYSUT1 DD DSN=FTP.RECEIVED.PDSABCD(MEMBERXY),
// DISP=SHR
//SYSUT2 DD DSN=USER1.FTP.RECEIVED.PDSABCD.MEMBERXY,
// DCB=(DSORG=PS,RECFM=FB,BLKSIZE=3120),
// UNIT=SYSDA,SPACE=(TRK,(5,5),RLSE),DISP=(NEW,CATLG,DELETE)
//SYSIN DD DUMMY
```

**Figure 13. Sample JCL Input Member after Replacement of Variables**

**Notes:**

1. Normally it is not necessary to replace the user ID and password in the job. When DVGJOBS submits a job, it is still running in the RACF® environment that was created for the file transfer. The submitted job will use the user ID and password of the RACF environment under which it is submitted.

2. When NetView FTP retrieves a password from RACF, the password is encrypted by RACF and cannot be used to replace a job variable. NetView FTP retrieves a password from RACF when the user specifies that NetView FTP is to do so.
How to Submit a Job That Has Just Been Transferred

You can use NetView FTP together with DVGJOBS to transfer a job to another system and, at the end of the transfer, to submit the job to the internal reader of that system.

For example, assume that your location is LOCA and the job that you want to transfer is contained in member JOBA of the PDS JOBS.AT.LOCA. You want to submit this job into the internal reader of the remote system LOCB. Before DVGJOBS can submit this job it must be stored into the PDS. This data set is to be called JOBS.AT.LOCA and should already exist at LOCB. The following batch-job control statements create such a file-transfer request and add it to the request queue:

```
FUNCTION=ADD
XMODE=TO
RMTNODE=LOCB
SFILEID='JOBS.AT.LOCA'
MEMBER=(R,JOBA)
RFILEID='JOBS.AT.LOCA'
PPXINFO='CR=(1,JOBS.AT.LOCA,JOBA)'
```

![Figure 14. Sample Batch-Job Control Statement to Create a File-Transfer Request](image)

The DVGEXJOB Sample Post-Transfer User-Exit Routine for NetView FTP VSE

DVGEXJOB is a sample post-transfer user-exit routine delivered on the installation tape. DVGEXJOB is in Assembler source format and can be used as supplied or can be modified to meet your needs. The functions of DVGEXJOB that are described in the following are already implemented in the shipped source code. You only have to assemble and link it.

What DVGEXJOB Does

You can use DVGEXJOB:

- On the sending system only
- On the receiving system only
- On both the sending and receiving systems.

DVGEXJOB releases one or more jobs at the end of the transfer. You must ensure that the jobs are released to a POWER queue. The originator of the file-transfer request can specify, in the request, the name of the power queue and the name of the job. The originator of the file-transfer request must be the same as the originator who puts the reusable jobs on the POWER queue. The originator of the file-transfer request is the user specified in the FCTORIG file-transfer request parameter or the name of the batch job. The originator who puts the reusable jobs on the POWER queue is the user ID specified in the POWER FROM card, or the ICCF user.

DVGEXJOB releases the specified job, or jobs, from the specified power queue.
How to Install DVGEXJOB

Before you can use DVGEXJOB, you must assemble and link-edit it to obtain a load module. During assembly, a library that contains the macro DVGAPL must be available to the assembler. DVGAPL.A is located on the installation tape.

All servers that use DVGEXJOB must be started with their PPEXIT initialization parameter set to one of the following:

(name,DVGEXJOB)

In this case, the transfer program uses name as the pre-transfer user-exit routine and DVGEXJOB as the post-transfer user-exit routine for all file transfers.

DVGEXJOB

In this case the transfer program uses DVGEXJOB as the pre-transfer and post-transfer user-exit routines for all file transfers. However, when DVGEXJOB is used as a pre-transfer user-exit routine, it performs no functions and immediately passes control back to the transfer program.

When a server that uses DVGEXJOB is started it loads DVGEXJOB. You must make sure that the library containing DVGEXJOB is available to the NetView FTP partition using the LIBDEF system control statement.

How to Invoke DVGEXJOB

A user can specify, in a file-transfer request, that a server uses DVGEXJOB as its post-transfer user-exit routine for the corresponding file transfer. The user does this by specifying an appropriate value for the User-Exit Routine Input parameter, which has the keyword PPXINFO. The value of this parameter also specifies how the server uses DVGEXJOB.

Syntax of the User-Exit Routine Input Parameter

DVGEXJOB works correctly only if you specify the value of the user-exit routine input parameter according to the following syntax:

parameter='type=(function,[jobloc],[job1],[job2],condition)]'

where:

parameter

This can be:

PPXINFO Batch job keyword
APLPPXIN Application program field name.

type

The system to which the job is submitted:

CS Sending system
CR Receiving system.
function
A code that specifies the function that a user-exit routine performs. When DVGEXJOB is used, specify the value 1. When you use a user-exit routine that performs some other function, or if you modify DVGEXJOB so that it performs some other function, use the following system to indicate which function it performs:

1  Job submission.
2  Data format conversion.
3  Authorization.
4  Encryption or decryption.
5  Memorization. This function is used internally by a user-exit routine to memorize the results of the processing of a file-transfer request.
6  Insertion of transfer parameters from a request library.
7  Write the receiving file to the spool queue.

All installations that implement these functions in their user-exit routines should follow this convention to ensure uniformity across the network. If you implement functions that are not members of this standard set, use nonnumeric codes, for the same reason.

jobloc
For MVS: Specifies the name of the PDS containing the jobs DVGEXJOB submits. If you omit this value, DVGEXJOB uses DVG.JOBLIB as the default.

For VSE: Specifies a VSE/POWER queue, where the job is released:

- RDR: VSE/POWER reader queue
- PUN: VSE/POWER punch queue
- LST: VSE/POWER list queue
- XMT: VSE/POWER xmit queue.

job1
The name of the member containing the job that DVGEXJOB submits if the file transfer is successful. A file transfer is successful if its return code is 8 or less. If you omit this value, DVGEXJOB does not submit a job if the file transfer is successful.

Note: The transfer program calls the post-transfer user-exit routine after all the main functions of a file transfer, including the transfer of all data and the closing of both data sets, has been completed. Any errors that occur after DVGEXJOB is called, for example, during the cleanup processing or user notification, do not affect the submission of the job.

job2
The name of the member containing the job that DVGEXJOB submits if the file transfer is unsuccessful or under the condition set by a status value. A file transfer is unsuccessful if its return code is more than 8. If you omit this value, DVGEXJOB does not submit a job if the file transfer is unsuccessful.
condition
A file-transfer reason code or the status of the file-transfer request. You can use the condition to control under which circumstances DVGEXJOB submits job2. This value can be specified only when job2 is specified, it does not trigger job1.

The condition can be one of the following:

reason code
job2 is submitted only if the decimal reason code from the file transfer matches this value.

A reason code for a file transfer is issued by the system at which the request was entered as well as by its partner system. Normally the reason code at the two systems is the same. However, in some situations they can be different, for example, when an error occurs at one system at which the request was entered as well as by its partner and the transfer program at that system does not have the opportunity to communicate the reason for the error to the transfer program at the other system.

status
job2 is submitted only if the status of the file-transfer request matches this value. The status of a file-transfer request at the end of a transfer can be one of the following:

W  The file-transfer request is waiting for automatic restart.
F  The file transfer finished, successfully or unsuccessfully, and is not automatically restarted.

The status of a file-transfer request is recorded by the requesting system and the responding system. Normally the statuses recorded by the two systems are the same. However, in some situations they can be different, for example, when an error occurs at one system and there is no chance for the transfer program at that system to communicate the new status to the transfer program at the other system. Only the status recorded by the requesting system is given to the request.

The following example shows you how to specify PPXINFO, so that:

- If the file transfer is successful, JOB1 is released from the reader queue of the sending system.
- If the file transfer is unsuccessful, JOB2 is released from the reader queue of the sending system.

PPXINFO='CS=(1,RDR,JOB1,JOB2)'
How DVGEXJOB Works in NetView FTP VSE

The sample NetView FTP VSE post-transfer user-exit routine releases one or more jobs from a VSE/POWER queue (RDR, PUN, LST or XMT) at the end of the file transfer. DVGEXJOB for NetView FTP VSE works as follows:

1. Returns immediately if not called as a post-transfer user-exit routine. Supplies a message but no return code.
2. Returns immediately if the file transfer did not finish successfully and job2 has not been specified. Supplies no message and no return code.
3. Checks for the correct specification of the exit routine input parameter.

   The format must be:
   CR=(1,VSE/POWER queue,job1,job2,condition)
   or
   CS=(1,VSE/POWER queue,job1,job2,condition)

   The VSE/POWER queue is optional. If this parameter is not specified, the VSE/POWER RDR queue is taken.
4. Sets up a cross-partition control block XPCCB (APPL=DVGEXJOB,TOAPPL=SYSPWR).
5. Issues an XPCC FUNC=IDENT to log on to the XPCC service.
6. Issues an XPCC FUNC=CONNECT to set up a communication path to VSE.
7. Sends a PWRSPCL FUNC=COMMAND,REQ=CTL,USERID=originator to issue either a PRELEASE queue,job or a PRELEASE queue,job2.
8. Issues an XPCC FUNC=DISCONN to disconnect the communication path to VSE.
9. Issues an XPCC FUNC=TERM to log off from the XPCC service.
10. Returns to the caller.

How DVGEXJOB Uses the Value of the Exit Routine Input Parameter

When the transfer program passes control to DVGEXJOB, DVGEXJOB scans the value of the User-Exit Routine Input parameter and determines what it should do:

- If the parameter is not specified, DVGEXJOB passes control back to the transfer program, without performing any functions.
- If the parameter is specified and is syntactically correct, DVGEXJOB attempts to carry out the specified function. If it is unable to carry out the function, DVGEXJOB writes an error message to the transfer program.
- If the parameter is specified but is syntactically incorrect, DVGEXJOB writes an error message to the transfer program.
Post-Transfer User-Exit Routine for NetView FTP VM to Run a REXX EXEC

The routine DVGEXEC in the sample library of the installation tape for your transfer program contains a sample post-transfer user-exit routine. DVGEXEC is supplied in assembler source format and can be used as supplied or modified to suit the needs of your installation. The sample NetView FTP VM post-transfer user-exit routine invokes one or more REXX EXECs at the end of a file transfer.

DVGEXEC for VM systems has basically the same function as DVGJOBS for MVS systems, the exception being that DVGEXEC runs at the same time as NetView FTP, whereas DVGJOBS runs independently of NetView FTP.

Installing DVGEXEC

Before you can use DVGEXEC you must assemble and link-edit it to obtain a load module. During assembly a macro library containing the macro DVGAPL must be available to the assembler. DVGAPL is on the installation tape in the sample library.

All servers that use DVGEXEC must be started with their PPEXIT initialization parameter set to one of the following:

(name,DVGEXEC)

In this case, the transfer program uses name as the pre-transfer user-exit routine and DVGEXEC as the post-transfer user-exit routine for all file transfers.

DVGEXEC

In this case, the transfer program uses DVGEXEC as the pre-transfer and post-transfer user-exit routines for all file transfers.

When a server that uses DVGEXEC is started it loads DVGEXEC from its load library.

Note: When coding the startup procedure for such a server, make sure that the load library in which DVGEXEC resides is accessible to the server.

Functions Performed by DVGEXEC

DVGEXEC can be used as a post-transfer user-exit routine for a file transfer on the sending system only, on the receiving system only, or on both the sending and receiving systems.

The following functions are performed when DVGEXEC is called as a post-transfer user-exit routine:

- Returns immediately if it is not called as a post-transfer user-exit routine. A message but no return code is supplied.
- Returns immediately if the file transfer has not yet finished successfully and jobname2 has not been specified.
- Checks for the correct specification of exit routine input parameter.
- Issues CMS SCAN and CMSCALL macros to invoke the EXECs named in jobname1 or jobname2 (depending on APLRC, APLRSN, and condition).
- Returns to caller.
Invoking DVGEXEC

A user can specify, in a file-transfer request, that a server uses DVGEXEC as its post-transfer user-exit routine for the corresponding file transfer. The user does this by specifying an appropriate value for the user-exit routine input parameter (keyword PPXINFO). The value of this parameter also specifies how the server uses DVGEXEC.

DVGEXEC works correctly only if you specify the value of the User-Exit Routine Input parameter according to the following syntax:

\[ \text{parameter} = 'CR=(1,,exec1,exec2,condition)' \]
\[ \text{parameter} = 'CS=(1,,exec1,exec2,condition)' \]
\[ \text{parameter} = 'CR=(1,,exec1,exec2,condition) \ CS=(1,,exec1,exec2,condition)' \]

where:

1  EXEC execution.
exec1 Name of the EXEC that runs if the file transfer is successful.
exec2 Name of the EXEC that runs if the file transfer is unsuccessful.

Refer to “Syntax of the User-Exit Routine Input Parameter” on page 67 for an explanation of the following variables:

- Parameter
- CR and CS
- Condition.

The following APL fields can be passed to the EXEC:

**Sending:**

APLSACUI, APLSACPW, APLSACGI, APLSDYFD, APLSDYLI, APLSDSS, APLSVOID, APLSVPW, APLSVEXT, APLSDIR

**Receiving:**

APLRACUI, APLRACPW, APLRACGI, APLRDYFD, APLRDYLI, APLRDSS, APLRVID, APLRVPW, APLRVEXT, APLRDIR

**Invoking DVGEXEC:**

```apl
/* REXX */
TRACE 0
SAY 'EXEC FDLCR1'
SAY 'Completion EXEC for receiving, file transfer ok'

/* FTP EXIT DVGJOBS provides following arguments */
SAY 'APLACUID = ' ARG(1)
SAY 'APLACPW = ' ARG(2)
SAY 'APLACGI = ' ARG(3)
SAY 'APLDYFD = ' ARG(4)
SAY 'APLDYLI = ' ARG(5)
SAY 'APLSS = ' ARG(6)
SAY 'APLVID = ' ARG(7)
SAY 'APLVPW = ' ARG(8)
SAY 'APLVEXT = ' ARG(9)
SAY 'APLDIR = ' ARG(10)

RETURN
```
Creating a Spool File Using the Pre- and Post-Transfer User-Exit Routines

Use one of the following sample routines to create a spool file:

- DVGSPPOOL for VM systems, described in “Using the Spool-File Routine DVGSPPOOL (NetView FTP VM)” on page 73.
- DVGXXMT for MVS systems, described in “Using the Spool-File Routine DVGXXMT (NetView FTP V2 MVS)” on page 75.

Using the Spool-File Routine DVGSPPOOL (NetView FTP VM)

The routine DVGSPPOOL in the sample library of the installation tape for your transfer program is a sample pre-transfer and post-transfer user-exit routine. DVGSPPOOL is supplied in Assembler source format, and can be used either as supplied or modified to suit the needs of your installation.

What DVGSPPOOL Does

You can use the DVGSPPOOL user-exit routine to create a spool file from the received file. It can only be called in the receiving system and must be invoked for both pre-transfer and post-transfer processing.

Figure 15 illustrates how DVGSPPOOL reads the records from a successfully transferred data set, and writes them to the system’s spool queue.

DVGSPPOOL attaches a tag to the spool file that identifies the owner of the spool file. The name of the owner was specified by the request originator using the User-Exit Routine Input parameter (the PPXINFO keyword for batch jobs, or the APLPPXIN field name for application programs). How DVGSPPOOL moves a data set to the spool queue and how you tag it depends on the operating system. When the spool file has been created the originally received data set is erased by DVGSPPOOL.
In addition to the functions shown in Figure 15 on page 73, DVGSPOOL checks whether it can convert the receiving file into a spool file. This is not possible for all file types, for example, VSAM. This check must be done before the transfer starts.

**Installing DVGSPOOL**

Before you can use DVGSPOOL, you must assemble and link-edit it to obtain a load module. During assembly, a macro library that contains the macro DVGAPL must be available to the assembler. DVGAPL is on the installation tape in the sample library.

All servers that use DVGSPOOL must be started with their PPEXIT server initialization parameter:

\[
\text{PPEXIT} = \text{DVGSPOOL or (DVGSPOOL,DVGSPOOL)}
\]

In this case the transfer program uses DVGSPOOL as the pre-transfer and post-transfer user-exit routines for all file transfers.

When a file server that uses DVGSPOOL is started, it loads DVGSPOOL from its load library. When coding the startup procedure for such a server, make sure that the load library in which DVGSPOOL resides is accessible to the server.

**Checks Performed by the Pre-Transfer User-Exit Routine:** The following checks are performed when DVGSPOOL is called as a pre-transfer user-exit routine:

- Return immediately if called in the sending system.
- Check if the request originator has specified the User-Exit Routine Input parameter.
- Check if the disposition indicates that the file already exists. Support is only provided for new files.
- Check if the type of the receiving file is supported by the spool file function. DVGSPOOL supports spooling for the following file types:
  - Physical sequential files on a DASD (CMS files)
  - Physical sequential files.

**Note:** If these checks are not invoked for pre-transfer processing, they are not performed.

**Functions Performed by the Post-Transfer User-Exit Routine:** The following functions are performed when DVGSPOOL is called as a post-transfer user-exit routine:

- Return immediately if the transfer has not finished successfully. This includes errors found at preparation time.
- Return immediately if called in the sending system.
- Check if the request originator has specified the user-exit input parameter.
- Allocate a punch file using the CMS FILEDEF command.
- Create a netdata header.
- Open the receiving file.
- Read the records from the receiving file.
- Write the netdata header plus the data records in netdata format to the virtual machine’s punch file.
- Close the receiving file.
- Tag the punch file and specify the destination of the spool file.
- Spool the punch file to RSCS.
- Close the punch file.
- Deallocate the punch file.
- Delete the receiving file using the CMS ERASE command.

**Invoking DVGSPOOL**

You can specify, in a file-transfer request, that a server is to use DVGSPOOL as its post-transfer user-exit routine for the corresponding file transfer. The user does this by specifying an appropriate value for the User-Exit Routine Input parameter, which has the keyword PPXINFO. The value of this parameter also specifies how the server uses DVGSPOOL.

DVGSPOOL works correctly only if you specify the value of the User-Exit Routine Input parameter according to the following syntax:

\[ parameter='CR=(7, \{userid\}, \{nodeid\}, \{condition\})' \]

where:

- **userid** User ID of the recipient of the spool file.
- **nodeid** Node ID of the recipient of the spool file. The nodeid is a character string of up to eight characters (contained in JES, VSE/POWER, RSCS).

Refer to “Syntax of the User-Exit Routine Input Parameter” on page 67 for an explanation of the following variables:

- Parameter
- CR
- Condition.

**Invoking DVGSPOOL:**

```
PPXINFO='CR=(7,userid,nodeid)'
```

**Using the Spool-File Routine DVGMXXMT (NetView FTP V2 MVS)**

The routine DVGMXXMT in the sample library of the installation tape for your transfer program is a sample pre-transfer and post-transfer user-exit routine. DVGMXXMT is supplied in assembler source format, and can be used as supplied or modified to suit the needs of your installation. DVGMXXMT applies to NetView FTP V2 MVS only.
What DVGMXXMT Does
You can use the DVGMXXMT user-exit routine to create a spool file from the received file. It can be called only in the receiving system.

Figure 16 illustrates how DVGMXXMT reads the records from a successfully transferred data set, and writes them to the system’s spool queue.

DVGMXXMT attaches a tag to the spool file that identifies the owner of the spool file. The name of the owner was specified by the request originator using the User-Exit Routine Input parameter. How DVGMXXMT moves a data set to the spool queue and how you tag it depends on the operating system. When the spool file has been created the originally received data set is erased by DVGMXXMT.

In addition to the functions shown in Figure 16 on page 76, DVGMXXMT checks whether it can convert the receiving file into a spool file. This is not possible for all file types, for example, VSAM and PDS. This check must be done before the transfer starts.

Installing DVGMXXMT
Before you can use DVGMXXMT, you must assemble and link-edit it to obtain a load module. During assembly, a macro library that contains the macro DVGAPL must be available to the assembler. DVGAPL is on the installation tape in the sample library.
All servers that use DVGMXXMT must be started with their PPEXIT (or POSTTRAN in the case of NetView FTP V2 MVS) server initialization parameter:

**PPEXIT=DVGMXXMT or (DVGMXXMT,DVGMXXMT)**

In this case the transfer program uses DVGMXXMT as the pre-transfer and post-transfer user-exit routines for all file transfers.

**PRETRAN=DVGMXXMT and POSTTRAN=DVGMXXMT**

When a server that uses DVGMXXMT is started, it loads DVGMXXMT from its load library. When coding the startup procedure for such a server, make sure that the load library in which DVGMXXMT resides is accessible to the server.

**Checks Performed by the Pre-Transfer User-Exit Routine:** The following checks are performed when DVGMXXMT is called as a pre-transfer user-exit routine:

- Return immediately if called in the sending system.
- Check if the request originator has specified the exit routine input parameter.
- Check if the disposition indicates that the file already exists. Support is only provided for new files.
- Check if the receiving file is dynamically allocated rather than allocated in the server job. Creation of spool files is only supported for dynamically allocated data sets.
- Check if the type of the receiving file is supported by the spool file function. DVGMXXMT supports spooling for the following file types:
  - Physical sequential files on a DASD
  - Physical sequential files on tape.

**Note:** These checks happen only if they are invoked for pre-transfer processing.

**Functions Performed by the Post-Transfer User-Exit Routine:** The following functions are performed when DVGMXXMT is called as a post-transfer user-exit routine:

- Return immediately if the transfer has not finished successfully. This includes errors found at preparation time.
- Return immediately if called in the sending system.
- Check if the request originator has specified the user-exit input parameter.
- Allocate a SYSOUT file using the dynamic allocation facility and specify the receiver of the spool file.
- Create a *netdata header*.
- Read the records from the receiving file.
- Write the *netdata header* plus the data records in netdata format to the SYSOUT file.
- Delete the receiving file.
**Invoking DVGMXXMT**

A user can specify, in a file-transfer request, that a server uses DVGMXXMT as its post-transfer user-exit routine for the corresponding file transfer. The user does this by specifying an appropriate value for the User-Exit Routine Input parameter, which has the keyword PPXINFO. The value of this parameter also specifies how the server uses DVGMXXMT.

DVGMXXMT works correctly only if you specify the value of the User-Exit Routine Input parameter according to the following syntax:

```
parameter='CR=(7,[userid],[nodeid],condition[])
```

where:

- **userid** User ID of the recipient of the spool file.
- **nodeid** Node ID of the recipient of the spool file. The *nodeid* is a character string of up to eight characters (contained in JES, VSE/POWER, RSCS).

Refer to “Syntax of the User-Exit Routine Input Parameter” on page 67 for an explanation of the following variables:

- Parameter
- CR
- Condition.

**Invoking DVGMXXMT:**

```
PPXINFO='CR=(7,userid,nodeid)'
```
Chapter 6. Writing a Post-Conversation User-Exit Routine

This chapter applies to NetView FTP V2 MVS only.

When a Post-Conversation User-Exit Routine Is Called

A post-conversation user-exit routine is only invoked on the local side, that is the side where the transfer was created.

A post-conversation user-exit routine is called (if specified by the server) immediately before the request’s status changes from active to not-active (which may be waiting or finished). A post-conversation user-exit routine is called even after unsuccessful attempts to establish a session with the partner.

The post-conversation user-exit routine is also called on abnormal termination of the server, unless:

- The abnormal termination occurred in either the pre-transfer or the post-transfer user-exit routines of this server or in the post-conversation user-exit routine itself.
- The server has been canceled by the operator.

A post-conversation user-exit routine is not called for a request that is going to be sent to the partner only to be checked. Checking of the request is requested via the Remote Check parameter. See the NetView FTP Parameter Reference for a description of this parameter.

The post-conversation user-exit routine is called even when the queue handler is no longer available. In this case, modifications made by the user-exit routine are not reflected on the transfer request queue.

Note: As the server must be linked to an APF authorized library, you must also link the user-exit routine to an APF authorized library.

What a Post-Conversation User-Exit Routine Can Do

A post-conversation user-exit routine can:

1. Under certain conditions control the further processing of the request.

2. Modify the user-exit input field APLPPXIN and the extended user-exit input. This field is pointed to by the field APLUXPTR.
   Modify the not-before and not-after times and dates (APLNBTIM, APLNATIM) and the transfer restart parameter APLREST. This makes sense when the post-conversation exit wants to change the status of the request from finished to waiting.

3. Provide messages.

Actions 1 and 2 on page 79 are controlled by the value the post-conversation user-exit routine puts into the APLRC field before returning control to the transfer program.
The following values are supported:

0   The user-exit routine does not want the status of the request to be modified, and no modifications have been made to the modifiable fields.

2   The user-exit routine does not want the status of the request to be modified, but modifications have been made to APLNBTIM, APLNATIM, APLREST, APLPPXIN, or to the extended user-exit routine input pointed to by APLUXPTR.

3   The user-exit routine wants the status of the request to be forced from finished to waiting. Additionally to this, modifications may have been made to one or more of the modifiable fields. This modification is only possible for unsuccessfully finished transfers, that is, transfers that have a return code greater than 4.

4   The user-exit routine wants the status of the request to be forced from waiting to finished.

Return codes other than 0, 2, 3, and 4 are not valid.

Note: A post-conversation user-exit routine should always reset the APLRC field to zero, if no action is taken. This is because this field is filled with the transfer return code on entry to the user-exit routine.

Changing Parameters

The post-conversation user-exit routine can insert values into the following APL fields:

- APLPPXIN
- APLUXLEN
- APLUXPTR

How the Transfer Program Handles Actions of the Post-Conversation User-Exit Routine

After the post-conversation user-exit routine returns control, the transfer program:

1. Sets the status of the request.
   
   The status of the request is changed from waiting to finished if the user-exit routine set a return code of 4 and the status would not have been finished anyway.
   
   A transfer reason code of 551 (RSNNOREQ) is set.
   
   Notification is done by issuing the transfer end message and by sending the report file (depends on the request parameters specified). If the user-exit routine provides messages, these are issued.

2. Requeues the request.
   
   The status of the request is forced from finished to waiting, if the user-exit routine set a return code of 3 and the status would not have been waiting anyway.
   
   A transfer reason code of 592 (RSNREQUE) is set.
   
   Action: Issue messages (see point 4) is performed.
3. Accepts modifications.

This action is only taken if the user-exit routine also sets a return code of 2 or 3.

Only user-exit routine input fields, not-before and not-after date and time fields, and the transfer restart parameter can be modified.

The contents of the modified date and time fields, and the restart parameter are checked for correctness. If their format is incorrect a message is issued and the status does not change from finished to waiting.

Action: Issue messages (see point 4) is performed.

4. Issues messages.

These messages are provided by the user-exit routine. The transfer program writes these messages to the transfer report file and to the server’s log file.

The default message (DVG110I) is issued if the user-exit routine does not provide messages but sets an unsupported return code value or a return code of 4. See Chapter 10, “Issuing Messages from User-Exit Routines” on page 103 for more information.

5. No action.

If the user-exit routine sets a return code of 0 the transfer program does not take further actions other than to issue messages.

Note: A request is regarded as “forced to status finished” only if its status was not finished anyway.

Considerations when Coding a Post-Conversation User-Exit Routine

You are recommended to consider the following when coding a post-conversation user-exit routine:

- The user-exit routine should provide an explanatory message whenever the status of the request is forced from waiting to finished.

- When forcing a request’s status from waiting to finished, or from finished to waiting, the transfer reason code (most likely set before) will be overwritten. To preserve information about the original reason for the failure, the user-exit routine should place this original reason code into a message.

- When forcing a request’s status from finished to waiting, the post-conversation user-exit routine should set a new not-after time in the APLNATIM, or reset the field to blank. This is because when the user-exit routine receives control, APLNATIM contains the transfer end time.

- Modifications to the user-exit routine input fields are normally preserved by the transfer program. However, this cannot be accomplished when, for example, the queue handler is no longer available when the request is about to be put back to the request queue.

- Global and local user-exit communication areas are available to the post-conversation user-exit routine only as input. The data contained in these areas is discarded immediately after the post-conversation user-exit routine has finished processing.
Chapter 7. Communicating with User-Exit Routines

This chapter describes what facilities NetView FTP provides to enable user-exit routines to communicate with each other.

Depending on the transfer program and the product version, user-specific information can be passed between user-exit routines. Figure 17 on page 83 shows the fields, the size (in bytes) of the data areas, and the transfer program to which they apply. Figure 18 on page 83 shows the user-exit routines and the transfer program to which they apply.

<table>
<thead>
<tr>
<th>Field Description</th>
<th>Number of Bytes</th>
<th>NetView FTP V2 MVS</th>
<th>NetView FTP VSE</th>
<th>NetView FTP VM</th>
<th>NetView FTP OS00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_1 for a User-Exit Routine</td>
<td>62</td>
<td>Can be specified and processed</td>
<td>Can be specified and processed</td>
<td>Can be specified and processed</td>
<td>Can be specified only</td>
</tr>
<tr>
<td>Input_2 for a User-Exit Routine</td>
<td>1200</td>
<td>Can be specified and processed</td>
<td>Can be specified only</td>
<td>Can be specified only</td>
<td>Can be specified only</td>
</tr>
<tr>
<td>Local Data Area</td>
<td>nnn</td>
<td>Can be specified and processed</td>
<td>Can be specified only</td>
<td>Can be specified only</td>
<td>Can be specified only</td>
</tr>
<tr>
<td>Global Data Area</td>
<td>30 000</td>
<td>Can be specified and processed</td>
<td>Can be specified only</td>
<td>Can be specified only</td>
<td>Can be specified only</td>
</tr>
</tbody>
</table>

The value nnn means that the size is restricted by the storage capacity of the system. The data areas are described in “Data Areas for Communicating between User-Exit Routines” on page 88.

<table>
<thead>
<tr>
<th>User-Exit Routine</th>
<th>NetView FTP V2 MVS</th>
<th>NetView FTP VSE</th>
<th>NetView FTP VM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-Queuing</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Pre-Transfer</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Post-Transfer</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Post-Conversation</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Note: NetView FTP/400 does not have user-exit routines. However, the input specified for user-exit routines at other NetView FTP systems can be passed to NetView FTP/400 where it can be processed differently.
Sequence of Calls to User-Exit Routines

The following rules apply to the sequence in which user-exit routines are called.\textsuperscript{13} The rules apply only to a normal transfer, that is not interrupted by a user-exit routine or by any other nonstandard situation. Exceptions from the rules are described later. Additionally, the following rules assume you have specified a name for each possible user exit.

1. It is always the pre-transfer user-exit routine at the sending system that is called first.
2. The next exit to be called is the pre-transfer user-exit routine at the receiving system.
3. The post-transfer user-exit routine at the sending system is called.
4. The post-transfer exit routine at the receiving system is the last of the user exits and has a counterpart at the sending system.
5. In NetView FTP V2 MVS the post-conversation user-exit routine is invoked at the requesting system, that is, the system where the file transfer request was created.

If normal processing is not interrupted by any exceptional situation (which includes temporary or permanent rejection of a transfer request by a pre-transfer user-exit routine), the above sequence is guaranteed. All of the exit routines run synchronized in the described sequence.

In this normal situation, each of the exit routines may modify some dedicated data areas in a way that the data areas can serve for communication between the different exit routines.

The normal calling sequence, described earlier is shown in Figure 25 on page 91 (for requests where the requesting system is the sending system) and in Figure 26 on page 92 (for requests where the responding system is the sending system).

Exceptions to the Calling Sequence

The sequence in which user-exit routines are called is influenced by exceptional situations that may occur during processing of a file-transfer request. For example:

- A pre-transfer user-exit routine temporarily rejects a file transfer request.
- A pre-transfer user-exit routine permanently rejects a file transfer request.
- NetView FTP detects or encounters an error, for example:
  - Invalid file-transfer request parameters or parameter values
  - Insufficient access authorization to files
  - Errors during access to files
  - Communication line failures.

\textsuperscript{13} A user-exit routine has no knowledge about which other user-exit routines are to be called at a later step in processing, and which function they are to perform. This is valid for user-exit routines running on the same system and for user-exit routines running on the partner system.

If a user-exit routine depends on the processing of a previously-called user-exit routine, a method must be established to let the user-exit routines communicate. This can be done by leaving a footprint in one of the data areas described in “Data Areas for Communicating between User-Exit Routines” on page 88.
Any of the described interruptions of the file transfer can result in a change of the calling sequence for user-exit routines. Additionally, the synchronization between different user-exit routines can be lost. The impact of interruptions on the calling sequence partially depends on the transfer direction also. Further descriptions are given for situations where:

- The interrupt occurs on the source system, when this is the requesting system (see Figure 19 on page 85)
- The interrupt occurs on the source system, when this is the responding system (see Figure 20 on page 86)
- The interrupt occurs on the target system, when this is the requesting system (see Figure 21 on page 87)
- The interrupt occurs on the target system, when this is the responding system (see Figure 22 on page 87).

### Figure 19. Call Sequence Exception. What is the impact on communication if a file-transfer request is interrupted on the source system when this is the requesting system?

<table>
<thead>
<tr>
<th>Point of Interruption</th>
<th>Resulting Impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>Before the target system is allowed to call the pre-transfer user-exit routine, for example:</td>
<td>• Pre-transfer user-exit routine on the target system is not called&lt;br&gt;• Post-transfer user-exit routine on the target system is called asynchronously (that is, not necessarily later than the corresponding one on the source system, and with no possibility to react to actions from the source system’s post-transfer user-exit routine)&lt;br&gt;• No possibility to communicate with the partner user-exit routines from the point of the interruption onwards.</td>
</tr>
<tr>
<td>• Open error on the source file&lt;br&gt;• Rejection by the pre-transfer user-exit routine.</td>
<td></td>
</tr>
<tr>
<td>After the target system is allowed to call the pre-transfer user-exit routine, but before the target system is allowed to call the post-transfer user-exit routine, for example:</td>
<td>• Post-transfer user-exit routine on the target system is called asynchronously (that is, not necessarily later than the corresponding one on the source system, and with no possibility to react to actions from the source system’s post-transfer user-exit routine)&lt;br&gt;• No possibility to communicate with the partner user-exit routines from the point of the interruption onwards.</td>
</tr>
<tr>
<td>• Error when sending data&lt;br&gt;• Error in the post-transfer user-exit routine.</td>
<td></td>
</tr>
<tr>
<td>After the target system is allowed to call the post-transfer user-exit routine, for example, if an error occurs in the post-conversation user-exit routine.</td>
<td>• No impact.</td>
</tr>
</tbody>
</table>

Chapter 7. Communicating with User-Exit Routines 83
**Figure 20. Call Sequence Exception. What is the impact on communication if a file-transfer request is interrupted on the source system when this is the responding system?**

<table>
<thead>
<tr>
<th>Point of Interruption</th>
<th>Resulting Impact</th>
</tr>
</thead>
</table>
| Before the target system is allowed to call the pre-transfer user-exit routine, for example: | • Pre-transfer user-exit routine on the target system is not called  
• Post-transfer user-exit routine on the target system is called asynchronously (that is, not necessarily later than the corresponding one on the source system, and with no possibility to react to actions from the source system’s post-transfer user-exit routine)  
• No possibility to communicate with the partner user-exit routines from the point of the interruption onwards. |
| • Open error on the source file  
• Rejection by pre-transfer user-exit routine. | |
| After the target system is allowed to call the pre-transfer user-exit routine, but before the target system is allowed to call the post-transfer user-exit routine, for example: | • Post-transfer user-exit routine on the target system is called asynchronously (that is, not necessarily later than the corresponding one on the source system, and with no possibility to react to actions from the source system’s post-transfer user-exit routine)  
• No possibility to communicate with the partner user-exit routine from the point of the interruption onwards. |
| • Error when sending data  
• Error in the post-transfer user-exit routine. | |
| After the target system is allowed to call the post-transfer user-exit routine, for example, when an error occurs when detaching the server subtask that handled the request. | • No impact. |
**Figure 21. Call Sequence Exception.** What is the impact on communication if a file-transfer request is interrupted on the target system when this is the requesting system?

<table>
<thead>
<tr>
<th>Point of Interruption</th>
<th>Resulting Impact</th>
</tr>
</thead>
</table>
| Before the source system is allowed to send data, for example: | • Post-transfer user-exit routine on the source system is called asynchronously (that is, not necessarily before the corresponding one on the target system)  
• No possibility to communicate with partner user-exit routines from the point of the interruption onwards. |
| • Rejection by pre-transfer user-exit routine  
• Open error on the target file. | |
| After the source system is allowed to send data, but before the target system is allowed to call the post-transfer user-exit routine, for example, when an error occurs when receiving data. | • Modifications from the pre-transfer user-exit routine of the target system are available on the source system  
• Post-transfer user-exit routine on the source system is called asynchronously (that is, not necessarily before the corresponding one on the target system)  
• No possibility to communicate with the partner user-exit routines from the point of the interruption onwards. |
| After the source system is allowed to call the post-transfer user-exit routine, for example, when an error occurs in the post-transfer user-exit routine on the target system. | • Information about the interruption is not available for the source system  
• Information from the source system’s post-transfer user-exit routine is not available on the target system. |

**Figure 22. Call Sequence Exception.** What is the impact on communication if a file-transfer request is interrupted on the target system when this is the responding system?

<table>
<thead>
<tr>
<th>Point of Interruption</th>
<th>Resulting Impact</th>
</tr>
</thead>
</table>
| Before the source system is allowed to send data, for example: | • Post-transfer user-exit routine on the source system is called asynchronously (that is, not necessarily before the corresponding one on the target system)  
• No possibility to communicate with the partner user-exit routines from the point of interruption onwards. |
| • Rejection by the pre-transfer user-exit routine  
• Open error on the target file. | |
| After the source system is allowed to send data, but before the target system is allowed to call the post-transfer user-exit routine, for example, when an error occurs when receiving the data. | • Modifications from the pre-transfer user-exit routine of the target system are available on the source system  
• Post-transfer user-exit routine on the source system is called asynchronously (that is, not necessarily before the corresponding one on the target system)  
• No possibility to communicate with the partner user-exit routines from the point of the interruption onwards. |
| After the source system is allowed to call the post-transfer user-exit routine, for example, when an error occurs in the post-transfer user-exit routine on the target system. | • Information from the target system’s post-transfer user-exit routine is not available on the source system. |
Data Areas for Communicating between User-Exit Routines

The data areas available for user-exit routines to communicate with each other are described as:

**Input_1**
Up to 62 bytes, the originator of a file transfer request can specify (with the parameter PPXINFO if the request is created via the batch job interface).

**Input_2**
Up to 1200 bytes, the originator of a file transfer request may specify if the responding system is NetView FTP V2 MVS (with the parameter PPXINFx if the request is created via the batch job interface).

For a description of how to specify these data areas, see the *User’s Guide* for your transfer program.

**Global Data Area**
Up to 30,000 bytes, created by a user-exit routine.

**Local Data Area**
Data area, created by a user-exit routine. NetView FTP V2 MVS does not control the length of this data area.

There are some restrictions as to which data area communicates to which user exit. For example, *local* data is never passed to the partner system. The data area restrictions are listed as follows:

**Input**
Data areas that can be specified by the originator of the file-transfer request and can serve as input to any user-exit routine.

**No_input**
Data areas that cannot be specified externally by the originator of the request but must be supplied by a user-exit routine.

**Permanent**
Data areas that are kept with the request and therefore available, for example, for a restart of the file transfer.

**Not_permanent**
Data areas that are never kept with the request and are therefore lost for a restart of the file transfer request.

**Passed**
Data areas that are passed to the partner of the file transfer on the other system.

**Not_passed**
Data areas that are passed to another user exit on the *same* system only, but are never sent to the partner system.

The assignment between data areas and attributes is shown in Figure 23 on page 88.

<table>
<thead>
<tr>
<th>Restrictions</th>
<th>Input_1</th>
<th>Input_2</th>
<th>Global</th>
<th>Local</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>No_input</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Permanent</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Not_permanent</td>
<td>-</td>
<td>-</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Passed</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
</tr>
<tr>
<td>Not_passed</td>
<td>✓</td>
<td>✓</td>
<td>-</td>
<td>✓</td>
</tr>
</tbody>
</table>
Where:

√ Restriction applies to the data area
— Restriction does not apply to the data area.

The flow of data areas between different user-exit routines is shown in Figure 25 on page 91 (for requests where the requesting system is the source system), and in Figure 26 on page 92 (for requests where the responding system is the source system).

What happens to the flow of data areas when a file-transfer request is interrupted, is described in more detail in “Exceptional Situations in Data Area Flow” on page 91.

### Rules for Modification of Data Areas

All of the data areas described in Figure 23 on page 88 can be modified by user-exit routines. However, there are some rules a user-exit routine must follow when modifying them.

1. Modifications are only accepted and preserved for the next user-exit routine in sequence, if the field APLRC indicates the modification by being set to the value of APLCHNGE. If this return code value is not set, modifications are lost.

2. The *Input_1* data area is always available in its full size of 62 bytes.

3. The *Input_2* data area is available only if specified during the creation of the transfer request, or if provided by any of the user-exit routines. If available, it will always have the length given by the last valid specification or modification, which may then be in a range from 1 to 1200 bytes.

   *Input_2* is *not* an area within the APL. It is a separate area. Only its address and length can be found in the APL (APLUXPTR and APLUXLEN). A user-exit routine may modify the contents of this area, as well as its length. The length of this area is always taken from the corresponding length field.

4. The *Local* area is available only if a user-exit routine creates it. Once created, it is passed to subsequent user-exit routines on the same system only. The *Local* area is never sent to the partner system.

   Subsequent user-exit routines that get this area may modify it in a similar way, as described for *Input_2*.

   The *Local* area is *not* an area within the APL, but a separate area. Only its address and length can be found in the APL (APLLXDPT and APCLXDLN).

5. The *Global* area is available only if a user-exit routine creates it. Once created, it is passed to subsequent user-exit routines on the same system and on the partner system.

   Subsequent user-exit routines that get this area passed to them, may modify it in a similar way, as described for *Input_2*.

---

14 Because this area is represented by an address and length, additional rules apply to its handling. These rules are described in Figure 24.
The Global area is not an area within the APL, but a separate area. Only its address and length can be found in the APL (APLGXDPT and APLGXDNL).

The additional rules that apply when modifying areas represented by an address and a length field are shown in Figure 24 on page 90.

**Figure 24. Rules for Modifying Communication Areas**

<table>
<thead>
<tr>
<th>Action</th>
<th>Address</th>
<th>Length</th>
<th>Considerations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change data only</td>
<td>Old address</td>
<td>Old length</td>
<td>The data, pointed to by Address, can be modified in any way that is appropriate to the user-exit routine.</td>
</tr>
<tr>
<td>Delete data</td>
<td>Set to X'0000'</td>
<td>Set to X'0000'</td>
<td>No storage must be freed by the user-exit routine. Cleanup processing is done by NetView FTP.</td>
</tr>
</tbody>
</table>
| Increase data length    | New address      | New length      | Increasing the data area must be done in the following sequence:  
1. Obtain storage with the new length  
2. Set address and length field in the APL to the new values  
3. Fill new area with data.  
The old storage area must not be freed by the user-exit routine. |
| New data area           | Address          | Length          | If a data area did not exist on entry, the user-exit routine needs to do the following:  
1. Obtain storage at the required size  
2. Set address and length field in the APL to the appropriate values  
3. Fill area with data. |
| Reduce data length      | [New|Old] address  | New length      | Reducing the size of a data area can be done by setting the new, smaller length value in the length field in the APL, or by the following:  
1. Obtain storage with the required size  
2. Set address and length field in the APL to the appropriate values  
3. Fill area with data.  
The old storage area must not be freed by the user-exit routine. |

**Note:** With the term **obtain storage**, the following must be considered:  
1. The corresponding address in the APL must point to the first byte of the obtained storage  
2. The corresponding length in the APL must give the complete length of the obtained area.
Exceptional Situations in Data Area Flow

The normal flow of data between user-exit routines can be interrupted by any of the events described in “Exceptions to the Calling Sequence” on page 84. If such exceptions occur, no further communication is possible between user-exit routines that are located on the partner system. The only indication that user-exit routines on the partner system receive about the exceptional situation is contained in the file-transfer reason code that can be found in the APLRSN field.

Calling Sequence and Data Flow

Figure 25 on page 91 and Figure 26 show the sequence in which user-exit routines are called, and the flow of data between those routines.

Figure 25. Calling User-Exit Routines. Sequence and data flow for requests where the requesting system is the source system.
Figure 26. Calling User-Exit Routines. Sequence and data flow for requests where the responding system is the source system.
Chapter 8. Considerations when Designing User-Exit Routines

This chapter describes the topics you need to consider when you design user-exit routines.

Accessing Data Sets Involved in a File Transfer

NetView FTP serializes access to the receiving data set. As a result, to ensure data integrity, concurrent writing the same receiving data set is not allowed. If you want to access the receiving data set in your user-exit routine, you must also force serialization with the same resource name NetView FTP uses. You can get the resource name NetView FTP enqueues with a tool such as RMF*:

The major and minor names are as follows:

**major : DVGrel/SV**
Where *rel* is the current product release.

**minor : volser : data set name**
The volume serial number the data set resides on and the name of the receiving data set.

Combining User-Exit Routines

You can integrate the pre-transfer user-exit routine, the post-transfer user-exit routine, and the post-conversation user-exit routine into a single module. Alternatively, you can replace the pre-transfer or the post-transfer user-exit routine by a security/statistics collection user-exit routine. The following combinations are possible:

- You specify **PPEXIT=FTPSECUR**.\(^\text{15}\)
  The transfer program calls the OPEN function of your routine at the beginning of a transfer and the CLOSE function at the end of a transfer.

- You specify **PPEXIT=ldmname**\(^\text{16}\) where *ldmname* is not FTPSECUR.
  The transfer program assumes that the pre-transfer and the post-transfer user-exit routines are included in one module. When the transfer program calls the module to run the pre-transfer user-exit routine, the APLVBC field contains the value APLPRERE. When the transfer program calls the module to run the post-transfer user-exit routine, the APLVBC field contains the value APLCOMRE.

- You specify **PPEXIT=(ldmname1,ldmname2)**, where *ldmname1* and *ldmname2* are different from each other, and both different from FTPSECUR.

\(^{15}\) FTPSECUR is the name of the security/statistics user-exit routine. See Chapter 9, “Writing a Security/Statistics Collection User-Exit Routine" on page 101.

\(^{16}\) In MVS and VM *ldmname* is the load module name, in VSE it is the phase name.
The transfer program assumes that the pre-transfer user-exit routine is contained in the first module, and the post-transfer user-exit routine is contained in the second module.

- You specify `PPEXIT=(FTPSECUR,Imdname)`, where `Imdname` is not FTPSECUR.
  
The transfer program calls the OPEN function of the FTPSECUR module at the beginning of a transfer, and treats the module `Imdname` as a post-transfer user-exit routine.

- You specify `PPEXIT=(Imdname,FTPSECUR)`, where `Imdname` is not FTPSECUR.
  
The transfer program treats the module `Imdname` as a pre-transfer user-exit routine, and calls the CLOSE function of the FTPSECUR module at the end of a transfer.

---

How the Pre-Transfer and Post-Transfer User-Exit Routines Work Together

The following example shows how the pre-transfer and post-transfer user-exit routines at the sending and receiving systems work together to transfer a data set whose type is not supported by the transfer programs:

1. Pre-transfer user-exit routine (sending): Unloads the data set to a temporary sequential data set and changes the values of the parameters of the sending data set so that the transfer program now transfers the temporary sequential data set.

2. Pre-transfer user-exit routine (receiving): Changes the values of the parameters of the receiving data set so that the transfer program now transfers the temporary sequential data set.

3. Post-transfer user-exit routine (sending): Deletes the temporary sequential data set (only if the transfer was successful).

4. Post-transfer user-exit routine (receiving): Converts the temporary data set to a data set of the same type as the original sending data set and then deletes the temporary data set (only if the transfer was successful).

---

Sequence of Calls for User-Exit Routines

**Note:** Case 1 and case 2 in the following apply to transfers using the LU 0, case 3 and case 4 apply to transfers using LU 6.2.

The sequence of calls depends on which system has obtained the first transfer request from the request queue.
Case 1: File-Transfer Request Is Obtained by the Sending Transfer Program (LU 6.2)

The NetView File Transfer Protocol defines the exchange of information between two transfer entities.

The NetView File Transfer Protocol supports two scenarios:

- A data transfer scenario to transfer data objects between the two request handlers involved.
- A request checking scenario with which a responding request handler checks the validity of a received request. No file transfer takes place.

Request handlers communicate by exchanging message units. The NetView File Transfer Protocol defines the message units and establishes the rules for sending them.

<table>
<thead>
<tr>
<th>Sending system</th>
<th>Receiving system</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-exit routine</td>
<td>Transfer program server</td>
</tr>
<tr>
<td>3. Send transfer request</td>
<td>5. Move parameters to DVGAPL and call EXIT</td>
</tr>
<tr>
<td>6. Pre-transfer exit</td>
<td></td>
</tr>
<tr>
<td>7. Check possibly changed parameters</td>
<td>8. Build temporary security environment</td>
</tr>
<tr>
<td>12. Send characteristics of the file to be sent</td>
<td>13. Extract sending file characteristics</td>
</tr>
<tr>
<td>13. Receive</td>
<td></td>
</tr>
</tbody>
</table>

Figure 27 (Part 1 of 3). Sequence of Calls for Pre-Transfer and Post-Transfer User-Exit Routines
<table>
<thead>
<tr>
<th>Sending system</th>
<th>Receiving system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User-exit routine</strong></td>
<td><strong>Transfer program server</strong></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>15. Move parameters to DVGAPL and call exit</td>
<td>16. Pre-transfer exit</td>
</tr>
<tr>
<td>17. Check possibly changed parameters</td>
<td></td>
</tr>
<tr>
<td>18. Build temporary security environment</td>
<td></td>
</tr>
<tr>
<td>19. Translate encryption key (MVS)</td>
<td></td>
</tr>
<tr>
<td>20. Allocate the file</td>
<td></td>
</tr>
<tr>
<td>21. Open the file</td>
<td></td>
</tr>
<tr>
<td>22. Send target information</td>
<td></td>
</tr>
<tr>
<td>23. Read data object</td>
<td></td>
</tr>
<tr>
<td>24. Compress and encypher data (MVS)</td>
<td></td>
</tr>
<tr>
<td>25. Send data object</td>
<td></td>
</tr>
<tr>
<td>28. If end-of-file, receive</td>
<td></td>
</tr>
<tr>
<td>30. Close the file</td>
<td></td>
</tr>
<tr>
<td>31. Move parameters to DVGAPL and call exit</td>
<td></td>
</tr>
<tr>
<td>32. Post-transfer exit</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 27 (Part 2 of 3). Sequence of Calls for Pre-Transfer and Post-Transfer User-Exit Routines*
### Figure 27 (Part 3 of 3). Sequence of Calls for Pre-Transfer and Post-Transfer User-Exit Routines

<table>
<thead>
<tr>
<th>Sending system</th>
<th>Receiving system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User-exit routine</strong></td>
<td><strong>Transfer program server</strong></td>
</tr>
<tr>
<td></td>
<td>35. Invoke post-transfer job submission (MVS)</td>
</tr>
<tr>
<td></td>
<td>36. Delete temporary security environment</td>
</tr>
<tr>
<td></td>
<td>37. Send post-transfer information</td>
</tr>
<tr>
<td></td>
<td>38. Receive</td>
</tr>
<tr>
<td></td>
<td>46. If NetView V2 MVS, deallocate the conversation</td>
</tr>
<tr>
<td></td>
<td>48. Move parameters to DVGAPL and call exit</td>
</tr>
<tr>
<td></td>
<td>50. Post-conversation exit</td>
</tr>
<tr>
<td></td>
<td>51. Set up statistical information</td>
</tr>
<tr>
<td></td>
<td>52. Notify the user</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note:** If step 2 in Figure 27 fails, the sending transfer program performs the steps 8, 35, and 36, and then continues with step 48.

If any of steps 6 to 12 in Figure 27 fail, the receiving transfer program continues with step 38, indicating in the APL that no file-transfer request is available (APLXMODE = blank). The sending transfer program continues with step 30, indicating in the APL that an error has occurred (APLRC=APLERROR).
Case 2: File-Transfer Request Is Obtained by the Receiving Transfer Program (LU 6.2)

Sending system

<table>
<thead>
<tr>
<th>User-exit routine</th>
<th>Transfer program server</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Obtain file transfer request from the Request Queue
2. Allocate a conversation
3. Confirm and receive
4. Send transfer request
5. Receive
6. Move parameters to DVGAPL and call EXIT
7. Pre-transfer exit
8. Check possibly changed parameters
9. Build temporary security environment
10. Generate encryption key (MVS)
11. Allocate file to be sent (MVS only: also retrieve data set information)
12. Open the file
13. Send characteristics of the file to be sent
14. Receive
14. Extract sending file characteristics
15. Build C/R key and check for existing C/R record
16. Move parameters to DVGAPL and call exit
17. Pre-transfer exit
18. Check possibly changed parameters
19. Build temporary security environment

Receiving system

<table>
<thead>
<tr>
<th>Transfer program server</th>
<th>User-exit routine</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 28 (Part 1 of 3). Sequence of Calls for Pre-Transfer and Post-Transfer User-Exit Routines
### Figure 28 (Part 2 of 3). Sequence of Calls for Pre-Transfer and Post-Transfer User-Exit Routines

<table>
<thead>
<tr>
<th>Sending system</th>
<th>Receiving system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User-exit routine</strong></td>
<td><strong>Transfer program server</strong></td>
</tr>
<tr>
<td>24. Read data object</td>
<td>24. Receive</td>
</tr>
<tr>
<td>25. Compress and encypher (MVS) data</td>
<td>27. Decypher (MVS) and decompress data</td>
</tr>
<tr>
<td>26. Send data object</td>
<td>28. Write data object</td>
</tr>
<tr>
<td>29. If end-of-file, receive</td>
<td>30. If no more data to be received send statistics report</td>
</tr>
<tr>
<td>31. Close the file</td>
<td>31. Receive</td>
</tr>
<tr>
<td>32. Move parameters to DVGAPL and call exit</td>
<td></td>
</tr>
<tr>
<td><strong>Post-transfer exit</strong></td>
<td></td>
</tr>
<tr>
<td>33. Post-transfer exit</td>
<td></td>
</tr>
<tr>
<td>34. Invoke post-transfer program call (MVS)</td>
<td></td>
</tr>
<tr>
<td>35. Deallocate the file</td>
<td></td>
</tr>
<tr>
<td>36. Invoke post-transfer job submission (MVS)</td>
<td></td>
</tr>
<tr>
<td>37. Delete temporary security environment</td>
<td></td>
</tr>
<tr>
<td>38. Send post-transfer information</td>
<td></td>
</tr>
<tr>
<td>39. Set up statistical information</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Receiving system</th>
<th>Sending system</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Transfer program server</strong></td>
<td><strong>User-exit routine</strong></td>
</tr>
<tr>
<td>20. Translate encryption key (MVS)</td>
<td></td>
</tr>
<tr>
<td>21. Allocate the file</td>
<td></td>
</tr>
<tr>
<td>22. Open the file</td>
<td></td>
</tr>
<tr>
<td>23. Send target information</td>
<td></td>
</tr>
<tr>
<td>24. Receive</td>
<td></td>
</tr>
<tr>
<td>27. Decypher (MVS) and decompress data</td>
<td></td>
</tr>
<tr>
<td>28. Write data object</td>
<td></td>
</tr>
<tr>
<td>30. If no more data to be received send statistics report</td>
<td></td>
</tr>
<tr>
<td>31. Receive</td>
<td></td>
</tr>
<tr>
<td>32. Move parameters to DVGAPL and call exit</td>
<td></td>
</tr>
<tr>
<td><strong>Post-transfer exit</strong></td>
<td></td>
</tr>
<tr>
<td>33. Post-transfer exit</td>
<td></td>
</tr>
<tr>
<td>34. Invoke post-transfer program call (MVS)</td>
<td></td>
</tr>
<tr>
<td>35. Deallocate the file</td>
<td></td>
</tr>
<tr>
<td>36. Invoke post-transfer job submission (MVS)</td>
<td></td>
</tr>
<tr>
<td>37. Delete temporary security environment</td>
<td></td>
</tr>
<tr>
<td>38. Send post-transfer information</td>
<td></td>
</tr>
<tr>
<td>39. Close the file</td>
<td></td>
</tr>
<tr>
<td>40. Move parameters to DVGAPL and call exit</td>
<td></td>
</tr>
<tr>
<td><strong>Post-transfer exit</strong></td>
<td></td>
</tr>
<tr>
<td>41. Post-transfer exit</td>
<td></td>
</tr>
<tr>
<td>42. Invoke post-transfer program call (MVS)</td>
<td></td>
</tr>
</tbody>
</table>
Figure 28 (Part 3 of 3). Sequence of Calls for Pre-Transfer and Post-Transfer User-Exit Routines

The following situations cause major deviations from the control flow shown in Figure 28:

1. If step 2 fails, the receiving transfer program performs the steps 19, 44, and 45, and then continues with step 48.

2. If any of steps 3 to 6 fail, the sending transfer program continues with step 32, indicating in the APL that no file-transfer request is available (APLXMODE = blank). The receiving transfer program continues with step 40, indicating in the APL that an error has occurred (APLRC=APLERROR).

3. If any of steps 8 to 14 fail, the sending transfer program continues with step 31. When it calls the post-transfer user-exit routine, it indicates that an error has occurred (APLRC=APLERROR). The receiving transfer program continues with step 40, indicating in the APL that an error has occurred (APLRC=APLERROR).
Chapter 9. Writing a Security/Statistics Collection User-Exit Routine

The FTP V2 security/statistics collection user exit is still supported by NetView FTP. The pre-transfer user-exit routine and the post-transfer user-exit routine have been designed to replace the FTP V2 security/statistics collection user-exit routine. These user-exit routines also provide an extended scope of functions. Only use the FTP V2 security/statistics collection user exit if you want to run your old FTP V2 exit routines, otherwise use the pre-transfer and the post-transfer user-exit routines.

The main difference between the security/statistics collection user-exit routine and the pre-transfer and post-transfer user-exit routines is the use of different control blocks used as an interface between the transfer program and a user-exit routine. The security/statistics collection user-exit routine uses the control block DVGSPL. The pre-transfer and post-transfer user-exit routines use DVGAPL.

The module name of the security/statistics collection user-exit routine must be FTPSECUR. Do not use this name for the name of the:

- Pre-transfer user-exit routine
- Post-transfer user-exit routine
- Post-conversation user-exit routine.

The name FTPSECUR causes the transfer program to prepare the DVGSPL control block instead of the DVGAPL control block.

Because the name of the original FTP V2 control block was changed from SUEPL to DVGSPL, you must recompile your security/statistics collection user-exit routine with the new control block. The field names remain unchanged.

If you want to run your existing security/statistics collection user-exit routine in combination with a new pre-transfer or post-transfer user-exit routine, see Chapter 8, “Considerations when Designing User-Exit Routines” on page 93. The functional scope of the security/statistics collection user exit remains unchanged from FTP V2.

**Note:** The CLOSE function of such a routine can be called without the OPEN function being called before. You can use Chapter 8, “Considerations when Designing User-Exit Routines” on page 93 to study the calling sequence for the security/statistics collection user-exit routine. In this case you can identify the pre-transfer user-exit routine with the OPEN function, and the post-transfer user-exit routine with the CLOSE function.
Chapter 10. Issuing Messages from User-Exit Routines

A transfer program can print the messages that a user-exit routine has written into an area after each call of the routine. The transfer program prints these messages when it regains control from the user-exit routine.

**Note:** The transfer program can print the messages only when the user-exit routine issues a valid return code.

The transfer program embeds a message from an exit routine into one of the following messages:

- Message DVG165I for a user-written file handler
- Message DVG111I for a pre-transfer, post-transfer, or a NetView FTP V2 MVS post-conversation user-exit routine.

The transfer program prints the messages in the transfer report file and the server log file (MVS and VM), or the report collection file (VSE).

Because the user-exit routines can issue messages through the transfer program's facilities after every function call, you can make a message closely related to the event that caused it to occur.

The Message Area

A user-exit routine must not put more than 1844 bytes into the message area, otherwise, the results are unpredictable. The 1844 bytes allow for ten 180-byte messages plus their number and length fields. A single message cannot be longer than 180 bytes.

The transfer program initializes the message area with X'00' before it calls the user-exit routine for the first time after a server is started. After the transfer program prints the messages following a function call, it initializes the message area back to X'00'.

The storage for the message area is allocated by the transfer program. It passes its address to the user-exit routines as described later in this chapter. If an error occurs during file transfer, NetView FTP places an error message into this area for use by the post-transfer and post-conversation user-exit routines. By setting the number of messages field to zero, NetView FTP does not treat this message as being set by the user-exit routine.

For a user-written file handler, the fourth word in the CALL argument list and the FUEMSGP field in the user file-handler parameter list point to the message area.
For a pre-transfer, post-transfer, or post-conversation user-exit routine, the APLMAPTR field in the APL points to the message area. The message area has the following layout:

<table>
<thead>
<tr>
<th>Byte position:</th>
<th>0</th>
<th>4</th>
<th>8............</th>
<th>n</th>
<th>(n+4).........</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of</td>
<td>Length of</td>
<td>Text of</td>
<td>Length of</td>
<td>Text of</td>
</tr>
<tr>
<td></td>
<td>messages</td>
<td>first</td>
<td>first</td>
<td>next</td>
<td>next</td>
</tr>
<tr>
<td></td>
<td>message</td>
<td>message</td>
<td>message</td>
<td>message</td>
<td>message</td>
</tr>
</tbody>
</table>

### Number of Messages

The first field identifies the number of messages that the user-exit routine stored in the message area during one function call and that the transfer program is to print. The transfer program can print a maximum of 10 messages. If the number of messages is greater than 10, the transfer program reduces the value to 10 and prints only the first ten messages.

If the value of this field is zero, the transfer program assumes that it is to print no messages after a function call. Before the transfer program calls a user-exit routine, it always sets this field to zero.

The transfer program treats this field as a 32-bit unsigned binary integer.

### Length of Message

The second field specifies the length of the first (or only) message that the transfer program is to print. The length specified must include the four bytes of the length field. So, the maximum length that you can specify is 184 bytes, but the transfer program prints only 180 bytes of the message text.

If a message is longer than 180 bytes, the transfer program truncates it. The transfer program indicates truncation by inserting three asterisks (*** into the last three bytes of the message.

If the binary value in the length field is not greater than 4, the transfer program issues an ABEND macroinstruction with a special abend code for user-exit error. For more information refer to *NetView FTP Messages and Codes*.

The transfer program treats this field as a 32-bit unsigned binary integer.

### Text of Message

The message text that the transfer program prints follows the length field. If the transfer program prints more than one message, another length field together with another message must follow the previous message text, and so on, up to a maximum of 10 messages.

The transfer program uses the length fields to skip from one message to the next. If, due to an incorrect length specification, the transfer program leaves out the message area, it reacts as when an incorrect length field is specified.
Messages for a Nonzero Return Code

If a user-exit routine issues a nonzero return code, the transfer program expects it to provide an error message in the message area. There are two exceptions to this rule:

- The GET function of a user-written file handler has set up the **no more records to transfer** return code. This return code indicates a normal end-of-file situation.

- A pre-transfer user-exit routine has indicated, by the APLCHNGE return code, that it has modified the values of the parameters in the APL.

Except for these two cases, if the transfer program finds no error messages in the message area, it supplies a substitute message. The substitute message can be either of the following:

- Message DVG166I for a user-written file handler
- Message DVG110I for a pre-transfer, post-transfer, or post-conversation user-exit routine.

These messages contain the return code from the exit routine.

When a session exists and a transfer is possible, the transfer program routes error messages indicating severe errors to the partner node. This ensures that the reason for the error is known to both nodes. If the user-exit routine supplies more than one error message, the transfer program transfers the last message in the message area to the partner node.\(^{17}\)

---

17 This restriction applies to transfers using the LU 0 protocol.
Chapter 11. Using the Trace Facility

The trace facility can be used with the following user exits:

- File-handler user exit
- Pre-transfer user exit
- Post-transfer user exit
- Post-conversation user exit (NetView FTP V2 MVS only).

The trace facility can be helpful when you are testing your exit routines.

This chapter describes what you can expect from the trace facility when you apply it to exit routines. For a full description of the trace facility and of trace classes, see the Installation, Operation, and Administration book for your transfer program. The concept of trace classes is used without further explanation in the following description.

There are four trace classes that you can use when you test your exit routines. These classes are 2, 3, 4, and 5. You use the trace class specification to control which kind of trace output you want to see and in which situation the trace is taken. You can use more than one trace class at a time.

The following list describes the different trace classes:

**Trace Class 2**
This type of trace applies only to the file-handler user-exit routine. With it, the transfer program traces the user file-handler parameter list before and after it calls the user-written file handler for the GET, PUT, or POINT functions. This trace shows how the file handler modifies the user file-handler parameter list.

The symptom record, which is dumped during the trace, contains message DVG220I.

**Trace Class 3**
This type of trace applies only to the user-written file handler. With it, the transfer program traces the storage area pointed to by FUETRCP in the length specified in FUETRCL, after it has called the file handler for the GET, PUT, or POINT functions.

The transfer program performs this trace if FUETRCL and FUETRCP contain positive binary values and if the high boundary of the storage area is higher than the low boundary. If FUETRCP together with FUETRCL do not denote a valid storage area, the results are unpredictable. After tracing, the transfer program resets FUETRCP and FUETRCL to binary zeros.

The symptom record, which is dumped during the trace, contains message DVG221I.
Trace Class 4
With this type of trace, the transfer program traces the appropriate parameter list before and after the user-exit routine is called by the transfer program.

The symptom record, which is dumped during the trace, contains message DVG220I.

User-written file handler:
The transfer program traces the user file-handler parameter list. The trace is taken only for functions OPEN and CLOSE.

Pre-transfer, post-transfer, and in NetView FTP V2 MVS post-conversation user exits:
The transfer program traces the APL.

Trace Class 5
With this type of trace, the transfer program traces an area defined by the user-exit routine after the transfer program has regained control from the user-exit routine.

The user-exit routine must specify in the appropriate parameter list both the address of the area that the transfer program is to trace and the length of this area.

The transfer program carries out this trace if the appropriate fields in the parameter list contain positive binary values and if the high boundary is higher than the low boundary of the storage area. If the parameter values do not denote a valid storage area, the results are unpredictable. After tracing, the transfer program resets these fields to zero.

The symptom record, which is dumped during the trace, contains message DVG221I.

User-written file handler:
The transfer program takes the trace only for the OPEN and CLOSE functions of the user-written file handler.

FUETRCP Points to the beginning of the area that the transfer program is to trace.

FUETRCL Contains the length of the area that the transfer program is to trace.

Pre-transfer, post-transfer, and post-conversation user exits:
APLPTRC Points to the beginning of the area that the transfer program is to trace.

APLPTRCL Contains the length of the area that the transfer program is to trace.
Invoking a Trace

You can invoke the four trace types individually, or in combinations, by specifying the appropriate trace statement. To invoke a trace in a:

**User-Written File Handler**
- For NetView FTP VM or NetView FTP VSE, specify a character in position 1 of the trace mask.
- For NetView FTP V2.2 MVS, specify module group 1 and one or more of the trace classes 2, 3, 4, and 5.
- These are the trace specifications for the module that calls the user-written file handler DVGCSUR.

**Pre-Transfer or Post-Transfer User-Exit Routines**
- For NetView FTP VM and NetView FTP VSE, specify a character in position 8 of the trace mask.
- For NetView FTP V2.2 MVS, specify module group 8 and one or both of the trace classes 4 and 5.
- These are the trace specifications for the module that calls pre-transfer and post-transfer user-exit routine DVGCSPP.

**Post-Conversation User-Exit Routine (NetView FTP V2 MVS only)**
- The module that calls the post-conversation user-exit routine is DVGMSPCX.

You activate the trace function by specifying the appropriate trace statements with the initialization parameter of the server (NetView FTP V2 MVS, NetView FTP VM) or the NetView FTP VSE partition.
Chapter 12. Using the APL

The Application Program Parameter List (APL), in conjunction with the APX for NetView FTP V2 MVS, contains all the parameters that enable a user-written program to communicate with a transfer program.

The DVGAPL macro is a PL/I and Assembler macro that provides mapping for the APL. The DVGCXCAP member in the ADVGSAC0 file on the distribution tape is also available for use in programs written in C language, and the external functions DVGCXRPT and DVGCXRGT for programs written in REXX. You can find a description of how to invoke the macro in Chapter 13, “Accessing the APL” on page 141.

The APL is used for:

**Application Program Interface**

You can specify values for parameters that make up a queue handler command, via the application program interface. The transfer program performs a validity and consistency check on the parameters specified and then performs the queue handler command. If it detects an error, it passes a return code and a reason code to the application program. For a detailed description about this process, see the User’s Guide for your transfer program.

**Pre-Queuing User-Exit Routine**

The transfer program invokes the pre-queuing user-exit routine after the parameters that accompany the queue handler command are validated. The transfer program passes the parameters of the APL to the pre-queuing user-exit routine, as they were specified by the application program interface routine, the EXEC interface routine (available with NetView FTP VM only), the batch job interface (not available with NetView FTP VM), or the interactive interface routine (not available with NetView FTP VSE). For a detailed description of the function of the pre-queuing user exit, see Chapter 3, “Writing a Pre-Queuing User-Exit Routine” on page 39.

**Pre-Transfer User-Exit Routine**

The transfer program invokes a pre-transfer user-exit routine before transferring any data. The transfer program passes control over the APL to the exit routine. The pre-transfer user exit can change certain parameters of the APL. For a more detailed description of the function of the pre-transfer user exit see Chapter 4, “Writing a Pre-Transfer User-Exit Routine” on page 45.

**Post-Transfer User-Exit Routine**

The transfer program invokes a post-transfer user-exit routine after the data is transferred. The transfer program passes control over the APL to the exit routine. The post-transfer user exit cannot change the values of any of the parameters of the APL. For a more detailed description of the function of the pre-transfer user exit see Chapter 5, “Writing a Post-Transfer User-Exit Routine” on page 57.
Post-Conversation User-Exit Routine

The transfer program invokes a post-conversation user-exit routine after the file transfer is completed. The post-conversation user-exit routine can change certain parameters of the APL. For a more detailed description of the function of the post-transfer user-exit routine see Chapter 6, “Writing a Post-Conversation User-Exit Routine” on page 79.

The APL provides fields for the following types of parameters:

- Parameters that an application program specifies when it issues an ADD, MODIFY, or any query or delete queue handler command.
- Parameters that an application program or a user-exit routine need to have in order to communicate with the transfer program or to make use of special functions that the transfer program offers for user programs.

The NetView FTP parameters are described in the NetView FTP Parameter Reference.

Indicated in this chapter are those parameters whose values can be changed by a user-exit routine. Any differences between using an APL field in an application program and in an exit routine are also described.

Layout of the APL

The installation tape for your transfer program contains a mapping module, DVGAPL, for the APL. Use this module as a reference to the data format of the parameters and for a complete list of all the fields.

On entering the user exits NetView FTP passes the APL with its initialized and specified values. If you have not specified values, character fields are initialized to X'40' (blank) and binary fields are initialized to X'00' (zero).

Identifying the APL

This section contains General-use Programming Interface and Associated Guidance information.

APLID (Identifier)

The APL Identifier must always contain the character string *DVGAPL* when communicating with the transfer program. This field serves the following main functions:

- Enables the transfer program to identify the APL.
- Shows the user-exit routines that the transfer program has correctly initialized the APL.
- Makes it easier to locate the APL in a dump.
If you are communicating with the transfer program through the application program interface, you must initialize the APL parameters to their proper values. As part of this process you must also set the APL identifier as one of the parameter values.

**Note:** The transfer program does not serve any requests unless APLID is set. If you are using a pre-queuing, pre-transfer, post-transfer, or post-conversation user-exit routine, the transfer program provides a correctly-initialized APL.

**APLVID (Version Identification)**
The version identification allows you to distinguish between different versions, releases, and modification levels. Each time a DVGAPL is created, APLVID and APLLNGTH must be filled from APLVIDC and APLLEN. If these fields are not filled, NetView FTP assumes that the user exit is intended to communicate with a different version of NetView FTP and does not copy the new fields.

The field APLVID is initialized to:
- `APLLVID21` For NetView FTP VM and NetView FTP VSE.
- `APLLVIA20` For NetView FTP V2.2.0 MVS.
- `APLLVIA21` For NetView FTP V2.2.1 MVS.

**APLLNGTH (APL Length)**
The APL length contains the total length of the APL control block.

---

**Feedback Fields and Verb Code**
This section contains General-use Programming Interface and Associated Guidance information.

**APLRC (Return Code)**
The return code shows the results of the processing that either the transfer program or a user program has performed.

When you pass the APL to the transfer program from either an application program or from a user-exit routine, you must initialize the APL return code to zero. In case of an error, set the appropriate return code as outlined in this book.

When the transfer program passes the APL to an application program, check the APL return code to determine whether an error has occurred during the execution of a queue handler command. For a detailed description of the possible return codes set by the transfer program, see *NetView FTP Messages and Codes*.

When the transfer program passes the control block to a pre-queuing user-exit routine, the APLRC field can have any of the following values:
- `APLQHCRX` Queue handler command rejected by pre-queuing user-exit routine.
- `APLCHNGE` APL modified by pre-queuing user-exit routine.

When the transfer program passes the control block to a pre-transfer user-exit routine, the APLRC field can have any of the following values:
- `APLCHNGE` APL modified by pre-queuing user-exit routine.
When the transfer program passes the APL control block to a post-transfer, or for NetView FTP V2 MVS only, a post-conversation user-exit routine, the APLRC field can have any of the following values:

- **APLOK**  Transfer successful.
- **APLTERM** Transfer not successful, error during termination.
- **APLError** Transfer not successful, error during transfer.
- **APLinier** Transfer not successful, error during initialization.
- **APLSever** Transfer not successful, severe process error.
- **APLABEND** Transfer ended with an abnormal termination.

**APLRSN (Reason Code)**

The APL reason code is used in conjunction with the APL return code to give more specific details about an error situation.

The reason code is not set by a user program but only by the transfer program when it passes the APL to a user program.

For a detailed description of the possible reason codes, see *NetView FTP Messages and Codes*.

**APLVBC (Verb Code)**

The APL verb code is used to specify the command or function to be performed. It can be set and interpreted both by the transfer program and a user program.

The following verb codes are valid in an application program, and are also passed to the pre-queuing user-exit routine by the transfer program:

- **APLADDRE**  ADD a request to the queue.
- **APLDELSR**  DELETE a specific request from the queue.
- **APLDELAR**  DELETE all requests, or all requests for a given originator.
- **APLDELFR**  DELETE all finished requests.
APLDELFO
FORCE deletion of an unsuccessfully finished request from the queue.

APLQRYSR
QUERY specific request.

APLQRYAR
QUERY all requests, or all requests for a given originator.

APLQRYAD
QUERY ADMINISTRATOR function.

APLMFYRQ
MODIFY any of the following request parameters:
Status
Server class
Request priority.

APLRSTRT
RESTART a specific request.

APLPRERE
Pre-transfer user-exit call.

APLCOMRE
Post-transfer user-exit call.

APLPTPRE
Post-transfer program call.

The following verb codes are set by the transfer program when it calls a user-exit routine:

APLPCXRE
Post-conversation user-exit call.

End of Applies to NetView FTP V2.2 MVS Only

End of Applies to NetView FTP V2 MVS Only

End of Applies to NetView FTP V2 MVS Only

End of Applies to NetView FTP V2 MVS Only
Applies to NetView FTP VSE Only

**APLRQREB**
Use in an application program to rebuild the request queue.

**APLPWCHG**
Use in an application program to change the master password.

End of Applies to NetView FTP VSE Only

Applies to NetView FTP V2.2 MVS Only

The following verb code is valid in an application program only:

**APLCLEAR**
INITIALIZE the fields of the APL except the fields APLVBC, APLLENGTH, and APLID. These fields must be set by the application program.

End of Applies to NetView FTP V2.2 MVS Only

Response and Message Area Fields

This section contains General-use Programming Interface and Associated Guidance information.

**APLFBAP (Address of QRA)**
Contains the address of an application program’s query response area (QRA). The specification is mandatory if an application program issues a QUERY queue handler command.

The transfer program uses this address to store the returned status information. For a detailed description of the structure of the user’s QRA, see the User’s Guide for your transfer program.

**APLFBALN (Length of QRA)**
Contains the length of the QRA (see: APLFBAP). The specification is mandatory if an application program issues a QUERY queue handler command.

**APLQACNT (Number of Status Items Placed in QRA)**
For QUERY commands from an application program, the transfer program uses this field to give information about the number of requests for which it could place status information into the program’s QRA (see: APLFBAP).

**APLQTTCNT (Number of Status Items Retrieved)**
For QUERY commands from an application program, the transfer program uses this field to give information about the number of requests for which it could retrieve status information.

**Note:** APLQTTCNT can be higher than or equal to APLQACNT, depending on the size of the application program’s QRA. If your QRA is too small for all the retrieved status information, the transfer program places the request number of the first request for which status information does not fit into the QRA in the APLRQNUM field. For a detailed description of this situation, see the User’s Guide for your transfer program.
**APLMAPTR (Pointer to Message Area)**

The message area is used to interchange messages between the transfer program and a user program. It cannot be used for a pre-queuing user-exit routine.

An application program can provide an area into which the transfer program inserts messages that it issued during its validity and consistency checking for a queue handler command. This pointer must be set before calling the transfer program. For a detailed description of this message facility, see the User’s Guide for your transfer program.

You can use this message area to pass messages between the transfer program and a pre-transfer, post-transfer, or post-conversation user-exit routine. In error situations, NetView FTP also uses this message area. The transfer program sets APLMAPTR when it passes control to the user-exit routine. For the structure of the message area, see Chapter 10, “Issuing Messages from User-Exit Routines” on page 103.

**APLMSGLN (Length of the Message Area)**

Contains the length of the message area (see APLMAPTR). An application program specifies the length of its message area. If the transfer program has issued messages, it passes as many messages as will fit into the message area. The transfer program specifies the length of the message area that it has allocated for use in a user-exit routine. For the structure of the message area, see Chapter 10, “Issuing Messages from User-Exit Routines” on page 103.

**APLMACT (Number of Messages in Message Area)**

Contains the number of messages that the transfer program has passed to an application program’s message area. It is only relevant for the application program interface, and is set by the transfer program.

When using the message area in an application program, this field must be initialized to zero before control is passed to the transfer program.

**APLMTOT (Total Number of Messages Available)**

Contains the total number of messages that the transfer program issued during the validation of the parameters and their values for a queue handler command. It is only relevant for the application program interface and is set by the transfer program.

When using the message area in an application program, this field must be initialized to zero before control is passed to the transfer program.

**Note:** APLMTOT can be greater than APLMACT if your message area is too small to contain all messages issued by the transfer program.

---

**Request Parameters**

This section contains General-use Programming Interface and Associated Guidance information.

**APLRQNUM (Request Number)**

For an ADD command from an application program this field must be initialized to zero before passing the APL to the transfer program. When the transfer program successfully performs the ADD command, it assigns a request number and returns its value in APLRQNUM.
For a QUERY, a RESTART, or a DELETE command from an application program, this field may or may not be set. If you refer to a specific file-transfer request, you must set APLRQNUM accordingly. If you refer to a group of requests, you only need to set APLRQNUM when you issue a QUERY command from a specific request number.

For a pre-queuing user-exit routine this field contains the value that the user specified. For an ADD queue handler command, this field is not relevant, as the transfer program has not yet set up the request number.

For a pre-transfer or post-transfer user-exit routine, this field contains the number of the currently processed request.

**APLCLASS (Server Class)**
When using an application program, you can assign a server class to the file-transfer request when carrying out any of the following queue handler commands:

ADD
MODIFY
QRYADM
RESTART.

A pre-queuing user-exit routine can change the file-transfer request’s class. For a more detailed description of the pre-queuing user exit see Chapter 3, “Writing a Pre-Queuing User-Exit Routine” on page 39.

When the transfer program passes the APL to the pre-transfer or post-transfer user-exit routines, APLCLASS contains the class of the current file-transfer request.

**APLPRTY (Request Priority)**
When using an application program to perform an ADD, RESTART, or MODIFY command you can assign a request priority to the file-transfer request.

A pre-queuing user-exit routine can change the file-transfer request’s priority. For a more detailed description of the pre-queuing user exit see Chapter 3, “Writing a Pre-Queuing User-Exit Routine” on page 39.

When the transfer program passes the APL to the pre-transfer or post-transfer user-exit routines, APLPRTY contains the priority of the current file-transfer request.

**APLORIG (Originator ID of Current Command)**
For each queue handler command that is issued through one of the user interfaces, the transfer program assigns an originator ID.

When the transfer program passes control to the pre-queuing, the pre-transfer, the post-transfer, or the post-conversation user-exit routines, it has already set up the APLORIG field.

**APLFORUS (Process Request for Another User)**
Tells the transfer program the originator ID of the request(s) that an application program wants to query, restart, delete, or modify.

It can be set up by an application program when it issues a QUERY, RESTART, DELETE, or MODIFY queue handler command and does not specify a request number.
Note: When APLFORUS is not specified for a QUERY ALL or DELETE FINISHED command, and a master password is specified instead, then the transfer program processes all file-transfer requests in the request queue.

APLPSWD (Password)
An application program can provide the password when it adds a file-transfer request to the request queue, or when it queries or deletes file-transfer requests.

A pre-queuing user-exit routine can change this field.

Refer to the User’s Guide for your transfer program to see when a password must be specified.

---

**APLPSWN (New password)**
This field changes the request administrator’s master password. You must also specify APLPSWD, the old password, when you specify APLPSWN.

Refer to the User’s Guide and the NetView FTP Parameter Reference for more information about passwords.

APLDELAY (Delay Time)
Contains the number of hours that must pass after a request has finished before it can be deleted.

---

Transfer Parameters
This section contains General-use Programming Interface and Associated Guidance information.

APLNBTIM (Not-Before Time and Date)
When the transfer program passes the value of this parameter to the pre-transfer user-exit routine, the value changes several times as the transfer program processes the file-transfer request. The only time the value of this parameter is the same as that specified by the request originator is when the file-transfer request is scheduled for the first time.

APLNATIM (Not-After Time and Date)
When the transfer program passes the value of this parameter to a pre-transfer user-exit routine, the value changes several times as the transfer program processes the file-transfer request. The only time the value of this parameter is the same as that specified by the request originator is when the file-transfer request is scheduled for the first time.

APLXMODE (Transfer Mode)
When the transfer program passes the Transfer Mode parameter to a pre-transfer or post-transfer user-exit routine, the value is not necessarily the same as in the file-transfer request, but it characterizes the location where the user-exit routine runs:

APLXMODE = APLTO Sending location
APLXMODE = APLFROM Receiving location.
When this field contains a blank it indicates that the transfer program has called a post-transfer user-exit routine but, due to an error, the file-transfer request was not yet available at that location.

**APLREQUE (Automatic Restart)**

This field can have either of the following values:

- **APLYES** Restart from checkpoint
- **APLNO** Restart from beginning of data set.

When the transfer program passes the value of this parameter to a pre-transfer, post-transfer, or post-conversation user-exit routine, it may have changed the original value from the file-transfer request. To enable a restart from the checkpoint in case of an automatic transfer restart, the transfer program sets the value of this parameter to **APLYES (Y)** after it has finished the initialization processing for the file transfer.

**APLREST (Restart from Checkpoint)**

This field can have either of the following values:

- **APLRLUNM** (Remote LU Name)
  - The pre-queuing user-exit routine can change, set, or blank out the remote LU name. If it changes or sets it, you must make sure that the server group name (APLNODE) is not set. If it blanks it out, you must make sure that the server group name (APLNODE) contains a valid server group.

- **APLRLUNM** (Remote LU Name)
  - The pre-queuing user-exit routine can change, set, or blank out the remote LU name. If it changes or sets it, you must make sure that the server group name (APLNODE) is not set. If it blanks it out, you must make sure that the server group name (APLNODE) contains a valid server group.

When the transfer program calls a pre-transfer user-exit or post-transfer user-exit routine, this field contains the LU name of the remote server with which the local server is in session.
APLNETID (Remote Network ID)
The pre-queuing user-exit routine can change, set, or blank out the remote network ID. If it changes or sets it, you must make sure that the server group name (APLNODE) is not set. If it blanks it out, you must make sure that the server group name (APLNODE) contains a valid server group.

When the transfer program calls a pre-transfer user-exit or post-transfer user-exit routine, this field contains the network ID of the remote AIX server with which the local server is in conversation.

APLLLUNM (Local LU Name)
The pre-queuing user-exit routine cannot modify the local LU name. When the transfer program calls a pre-transfer or post-transfer user-exit routine, this field contains the LU name of the local server that processes the file-transfer request.

APLRWSID (Remote Workstation Address or Remote Workstation Nickname)
Either the decimal representation of the absolute IP network address and the subnet address of the remote AIX workstation or the locally defined nickname for the network ID and the subnet address of the remote workstation.

APLPPXIN (Input for User-Exit Routine)
Modifications to the user-exit routine input field in the pre-transfer user-exit routine of the sending system are passed to the pre-transfer user-exit routine on the receiving system, which can again modify it. After the receiving pre-transfer user-exit routine has modified the value of this parameter, the post-conversation user-exit routine at the requesting system may modify it before the transfer program updates the file-transfer request in the request queue to reflect the changes.

APLUXPTR (Address)
The address of the user-exit routine input field.

APLUXLEN (Length)
The length of the user-exit routine input field.

APLAPUID (APPC Conversation User ID)
User ID required by a NetView FTP/400 or NetView FTP/2 system.

APLAPPWD (APPC Conversation Password)
Password required by a NetView FTP/400 or NetView FTP for Workstations system.

APLRJPDS (Post-Transfer Jobs)
The name of the PDS library containing the jobs to be submitted.
APLRJOKn
The names of jobs to be submitted after a successful transfer.

APLRJNOn
The names of jobs to be submitted after an unsuccessful transfer.

APLRPPTR
Address of the post-transfer program data.

APLRPLEN
Length of the post-transfer program data.

APLRPNAM (Post-Transfer Program Name)
Contains the name of the program to be invoked after the transfer has finished.

APLRRCHK (Remote Check Option)

Recipients of the Reports
This section contains General-use Programming Interface and Associated Guidance information.

APLSNUID (User ID for Program at Sending System)
Contains the user ID of the recipient of the report, sent by the transfer program at the sending node.

APLSNNID (Node ID for Program at Sending System)
Contains the JES node name, the RSCS node ID, or the VSE/POWER node name of the recipient of the report, sent by the transfer program at the sending node. When a user program specifies a value for this parameter, it must also specify a value for APLSNUID.

APLRNUID (User ID for Program at Receiving System)
Contains the user ID of the recipient of the report, sent by the transfer program at the receiving node.

APLRNNID (Node ID for Program at Receiving System)
Contains the JES node name, the RSCS node ID, or the VSE/POWER node name of the recipient of the report, sent by the transfer program at the receiving node. When a user program specifies a value for this parameter, it must also specify a value for APLRNUID.

File Parameters: User-Written File Handler Input
This section contains General-use Programming Interface and Associated Guidance information.

APLUEUI0 to APLUEUI9 (File Handler Input)
A pre-transfer user-exit routine can modify the user-written file handler input parameters if:

- The data set type (APLSFTYP and APLRFTYP) is APLUSER for NetView FTP VM and NetView FTP VSE.
- The file organization (APLSFORG and APLRFORG) is APLUSER for NetView FTP V2.2 MVS.
Modifications to these fields in the pre-transfer user-exit routine of the sending node are passed to the pre-transfer user-exit routine of the receiving node, which can again modify them.

After the receiving pre-transfer user-exit routine has modified the values of the parameters, the transfer program updates the file-transfer request on the request queue to reflect the changes.

File Parameters: Sending File

This section contains General-use Programming Interface and Associated Guidance information.

The following lists the parameters that apply to the sending file.

Security Parameters

APLSACUI (User ID of up to 8 characters)

| Applies to NetView FTP V2.2 MVS Only |

APLSACU2 (User ID of up to 10 characters)

| End of Applies to NetView FTP V2.2 MVS Only |

APLSACPW (Password)

APLSACGI (Group ID)

When some or all of the values for the security parameters are retrieved by the transfer program, the retrieved values are passed to the pre-transfer, the post-transfer, and post-conversation user-exit routines. The parameter values specified by the originator of the request are passed to the pre-queuing user-exit routine only.

Retrieved passwords are encrypted and the exit routines cannot read them.

Although the pre-transfer user-exit routine at the sending system can change security parameters, it cannot request the transfer program to retrieve them. So, if the pre-transfer user-exit routine specifies an asterisk (*) for a security parameter, the transfer program does not retrieve them again.

APLSENCR (RACF Encryption)

If YES is specified for this field, NetView FTP assumes that the password specified for the sending system’s security parameters is RACF encrypted.
File Information

APLSDYFD (Data Set Name or File ID)

APLSDYMM (PDS Member Name)
If the file-transfer request or the pre-transfer user-exit routine does not specify a name for the receiving file, the transfer program retrieves one from the system control blocks (job allocation). This name is presented to the post-transfer and the post-conversation (NetView FTP V2 MVS only) user-exit routine. If the sending data set is a member of a PDS and job allocation is used, this field does not contain the member name.

This field contains only a value if this was specified in the file-transfer request (dynamic allocation case).

APLSDYDD (MVS DD Name or VSE File Name)
When the transfer program calls a pre-transfer, post-transfer, or post-conversation user-exit routine, it inserts in this field the DD name that it uses to allocate the sending data set. This is true for job allocation, where the transfer program uses the DD name from the file-transfer request, and also for dynamic allocation, where the transfer program creates its own DD name (DVGFIxx).

APLSDYUC (VSAM Catalog Name)

APLSDVPW (VSAM Cluster Password)

Does Not Apply to NetView FTP V2.2 MVS

APLSFTYP (File Type)
One of the following values:

APLLAB Data set on labeled tape
APLUNLAB Data set on unlabeled tape
APLPO Partitioned data set
APLPS Physical sequential data set
APLUSER User-handled data set
APLVSAM VSAM cluster.

A pre-transfer user-exit routine can change this parameter. However, it cannot change it to APLUSER, and it cannot change it if it was originally set to APLUSER in the file-transfer request.

End of Does Not Apply to NetView FTP V2.2 MVS

APLSDYRE (Record Format)
One of the following values:

APLUND Undefined record format
APLVAR Variable length records (unblocked)
APLFIX Fixed length records (unblocked)
APLFXBL Fixed length records (blocked)
APLVARBL Variable length records (blocked)
APLVARNS  Variable spanned
APLVARBS  Variable blocked spanned.

When a user program specifies this parameter, it must also specify values for
APLSDYBL and APLSDYLR.

APLSDYBL (Block Size)
When a user program specifies a value for this parameter, it must also specify
values for APLSDYRE and APLSDYLR.

APLSDYLR (Logical Record Length)
When a user program specifies this parameter, it must also specify values for
APLSDYRE and APLSDYBL.

APLSDYUT (Device Type - Unit)

APLSDYLI (List of Volume Serial Names)
When a user program specifies one or more volume serial numbers here, it
must ensure that each volume serial number has a length of six characters.
Each new number must start on a 6-byte boundary. When the transfer
program calls a pre-transfer, post-transfer, or post-conversation user-exit
routine, the first entry in this field can contain a number that the transfer
program has retrieved from system control blocks. This number is not neces-
sarily the same as the number that was specified in the file-transfer request.

Applies to NetView FTP VSE Only

APLSDYQU (Quote Indicator)
One of the following values:

APLYES  The transfer program treats the first volume serial number as
         if specified in quotes.
APLNO   The transfer program treats the first volume serial number as
         if not specified in quotes.

End of Applies to NetView FTP VSE Only

APLSDYPH (Load Module Name of a User-Written File Handler)

Applies to NetView FTP VSE Only

APLPSWD (Password)

APLPSWN (New Password)

APLDELAY (delay time)
Use in an application program to define a time delay before rebuilding the
request queue.

End of Applies to NetView FTP VSE Only
APLSACCO (File Access Option)

APLSEOPS (End-of-Processing Option)

APLSCCSI (Coded Character Set Identifier)

APLSFORG (File Organization)

One of the following values:

- **APLPO**  Partitioned data set
- **APLPS**  Physical sequential data set
- **APLUSER**  User-handled data set
- **APLVSAM**  VSAM cluster.

A pre-transfer user-exit routine can change this parameter. However, it cannot change it to APLUSER, and it cannot change it if it was originally set to APLUSER in the file-transfer request.

APLSFLAB (Label Type)

APLSLFA (Address of Sending File ID)

APLSLFL (Length of Sending File ID)

APLSFHMD (File Handling Mode)

APLSEOR (Record Delimiter)

APLSEOF (End-of-File Option)
VM Link Parameters

APLSVDID (Disk User ID)
A user program must specify a value for this parameter.

APLSVEXT (Disk Address)
When a user program specifies a value for this parameter, it must also specify a value for APLSVDID.

APLSVINT (Internal Disk Address)
When a user program specifies this parameter, it must also specify values for APLSVDID, APLSVEXT, and APLSVFML, but not for APLSVPW.

APLSVFML (Internal Filemode)
When a user program specifies this parameter, it must also specify values for APLSVDID, APLSVEXT, and APLSVINT, but not for APLSVPW.

APLSVPW (Link Password)
When a user program specifies a value for this parameter, it must also specify values for APLSVDID and APLSVEXT, but not for APLSVINT and APLSVFML.

SFS Parameters

APLSPOID (Pool ID)
Pool ID of the SFS pool containing the sending file.

APLSOWID (Top Directory)
User ID who owns the directory containing the sending file.

APLSSUD1...8 (Subdirectory Level)
Names of the subdirectories that contain the sending file.

OS/400 Parameters

APLS4LIB (OS/400 Library Name)

APLS4FIL (OS/400 File Name)

APLS4MBR (OS/400 Member Name)

APLS4OTY (OS/400 File Type)
File Parameters: Receiving File

This section contains General-use Programming Interface and Associated Guidance information.

The following lists the parameters that apply to the receiving file:

Security Parameters

APLRACUI (User ID of up to 8 characters)

[Applies to NetView FTP V2.2 MVS Only]

APLRACU2 (User ID of up to 10 characters)

[End of Applies to NetView FTP V2.2 MVS Only]

APLRACPW (Password)

APLRACGI (Group ID)

When some or all of the values for the security parameters are retrieved by the transfer program, the retrieved values are passed to the pre-transfer, post-transfer, and post-conversation user-exit routines. The parameter values specified by the originator of the request are passed to the pre-queuing user-exit routine only.

Retrieved passwords are encrypted and the exit routines cannot read them.

Although the pre-transfer user-exit routine at the sending system can change security parameters, it cannot request the transfer program to retrieve them. So, if the pre-transfer user-exit routine specifies an asterisk (*) for a security parameter, the transfer program does not retrieve them again.

APLRENCR (RACF Encryption)

If YES is specified for this field, NetView FTP assumes that the password specified for the receiving system’s security parameters is RACF encrypted.

File Information

APLRDYFD (Data Set Name or File ID)

If the file-transfer request or the pre-transfer user-exit routine does not specify a name for the receiving file, the transfer program retrieves one from the system control blocks (job allocation) or creates one by itself (dynamic allocation) after the call of the pre-transfer user-exit routine. This name is also presented to the post-transfer and the post-conversation (NetView FTP V2 MVS only) user-exit routine.

APLRDYMM (PDS Member Name)

If the receiving data set is a member of a PDS and job allocation is used, then this field does not contain the member name.

This field contains a value only if this was specified in the file-transfer request (dynamic allocation case).
APLRDYDD (MVS DD Name or VSE File Name)
When the transfer program calls a pre-transfer, post-transfer, or post-conversation user-exit routine, it inserts in this field the DD name that it uses to allocate the receiving file. This is true for job allocation, where the transfer program uses the DD name from the file-transfer request, and also for dynamic allocation, where the transfer program creates its own DD name (DVGFI01 to DVGFI32 for MVS, DVGFI00 to DVGFI0F for VSE).

APLRDYUC (VSAM Catalog Name)

APLRDYKF (Processing Option for VSAM KSDS)
One of the following values:
- APLADDK: Add to data set by key
- APLADDB: Add to beginning of data set
- APLDELIK: Delete by key
- APLREP: Replace by key
- APLMERGE: Merge into data set

Does Not Apply to NetView FTP V2.2 MVS

APLRFTYP (File Type)
One of the following values:
- APLLALB: Data set on labeled tape
- APLUNLAB: Data set on unlabeled tape
- APLPO: Partitioned data set
- APLPS: Physical sequential data set
- APLUSER: User-handled data set
- APLVSAM: VSAM cluster.

A pre-transfer user-exit routine can change this parameter. However, it cannot change it to APLUSER, and it cannot change it if it was originally set to APLUSER in the file-transfer request.

End of Does Not Apply to NetView FTP V2.2 MVS

APLRDYRE (Record Format)
This field can have any of the following values:
- APLUND: Undefined record format
- APLVAR: Variable length records (unblocked)
- APLFIX: Fixed length records (unblocked)
- APLFIXBL: Fixed length records (blocked)
- APLVARBL: Variable length records (blocked)
- APLVARSP: Variable spanned
- APLVARBS: Variable blocked spanned.

When a user program specifies a value for this parameter, it must also specify values for APLRDYBL and APLRDYLR.
APLRDSS (Data Set Sequence Number)

APLRDYBL (Block Size)
When a user program specifies a value for this parameter, it must also specify values for APLRDYRE and APLRDYLR.

APLRDYLR (Logical Record Length)
When a user program specifies this parameter, it must also specify values for APLRDYRE and APLRDYBL.

When a user program uses this parameter to specify the maximum record size of a VSAM cluster, it must also specify a value for APLDYRSA.

APLRSPRM (Primary Disk Space Quantity)
When a user program specifies this parameter, it must also specify a value for APLRSUNI.

APLSSESEC (Secondary Disk Space Quantity)
When a user program specifies this parameter, it must also specify values for APLRSUNI and APLSPRM.

APLDRDIRB (Number of Directory Blocks)
Number of directory blocks allocated for the receiving file.

APLRAVBL (Average Block Length Block Amounts)
When a user program specifies this parameter, it must also specify the value APLBLK for APLRSUNI, and a value for APLRSPRM.

APLRSUNI (Space Unit)
One of the following values:

- **APLCYL**: If cylinders are allocated
- **APLTRK**: If tracks are allocated
- **APLBLK**: If blocks are allocated
- **APLREC**: If records are allocated (in NetView FTP V2 MVS) or if VSAM space is allocated in records.

Does Not Apply to NetView FTP V2.2 MVS

APLRDYTD (Output Disposition)
One of the following values:

- **APLNEW**: DISP=(NEW,KEEP,KEEP)
- **APLOLD**: DISP=OLD
- **APLCAT**: DISP=(NEW,CATLG,KEEP)
- **APLSHR**: DISP=SHR
- **APLMOD**: DISP=MOD
- **APLREGL**: DISP=REG.

End of Does Not Apply to NetView FTP V2.2 MVS
APLRDYDN (Output Tape Density)
One of the following values:

'0' Density 200 bpi
'1' Density 556 bpi
'2' Density 800 bpi
'3' Density 1600 bpi
'4' Density 6250 bpi
'C' Density 38912 bpi (VM and VSE only).

APLRDYEX (Expiration Date or Retention Period)
Expiration date or retention period of a file member. If the requesting system is NetView FTP/400, the date can be a year after and including the year 2000.

APLRDYLs (VSE Logical Record Size)
The logical record size is used only for new receiving SAM files in VSAM managed space (SAM ESDS) in a VSE system. Only specify a logical record size for a SAM file in VSAM managed space with a fixed blocked record format. The logical record size can be from 1 to 32761 bytes long.

APLRDYUT (Device Type - Unit)
Device type of a unit on which a file is stored.

APLRVOC (Volume Count)
Maximum number of volumes on which a receiving file can reside.

APLRDYLI (List of Volume Serial Names)
When a user program sets one or more volume serial numbers here, it must ensure that each volume serial number has a length of six characters. Each new number must start on a 6-byte boundary.

When the transfer program calls a pre-transfer, post-transfer, or post-conversation user-exit routine, the first entry in this field can contain a number that the transfer program has retrieved from system control blocks. This number is not necessarily the same as the number that was specified in the file-transfer request.

Applies to NetView FTP VSE Only

APLRDYQU (Quote Indicator)
One of the following values:

APLYES The transfer program treats the first volume serial number as if specified in quotes.

APLNO The transfer program treats the first volume serial number as if not specified in quotes.

End of Applies to NetView FTP VSE Only

APLRDYPH (Load Module Name of a User-Written File Handler)
User-written file handler name containing the load module name (MVS, VM) or phase name (VSE) of the user-written file handler that gains access to the file.
APLRPADD (Padding Character)

APLRSOPT (File Status Option)

APLRACCO (File Access Option)

APLRPROC (File Processing Option)

APLREOPS (End-of-Processing Option)

APLRCCSI (Coded Character Set Identifier)

APLRFORG (File Organization)

One of the following values:

- APLPO  Partitioned data set
- APLPS  Physical sequential data set
- APLUSER  User-handled data set
- APLVSAM  VSAM cluster.

A pre-transfer user-exit routine can change this parameter. However, it cannot change it to APLUSER, and it cannot change it if it was originally set to APLUSER in the file-transfer request.

APLRFLAB (Label Type)

APRLLFA (Address of Sending File ID)

APRLFL (Length of Sending File ID)

APLRFHMD (File Handling Mode)

APLREOF (End-of-File Option)

APLRHIDA (Hidden-File Attribute)

APLRSYSA (System-File Attribute)

APLRRDOA (Read-Only File Attribute)

APLRARCA (Archive-File Attribute)
VM Link Parameters

APLRVDID (Disk User ID)

APLRVEXT (Disk Address)
When a user program specifies a value for this parameter, it must also specify a value for APLRVDID.

APLVINT (Internal Disk Address)
When a user program specifies a value for this parameter, it must also specify values for APLRVDID, APLRVEXT, and APLRVFML, but not for APLRVPW.

APLRVFML (Internal Filemode)
When a user program specifies a value for this parameter, it must also specify values for APLRVDID, APLRVEXT, and APLRVINT, but not for APLRVPW.

APLRVPW (Link Password)
When a user program specifies a value for this parameter, it must also specify values for APLRVDID and APLRVEXT, but not for either APLRVINT or APLRVFML.

SFS Parameters

APLRPOID (Pool ID)
Pool ID of the SFS pool containing the receiving file.

APLROWID (Top Directory)
User ID that owns the directory containing the receiving file.

APLRSUD1...8 (Subdirectory Level)
The names of the subdirectories that contain the receiving file.

SMS Parameters

APLRSAVG (Average Record Multiplier)
One of the following values for the number of records:

APLUNIT Units
APLKILO Thousands
APLMEGA Millions.

APLRSDAT (Data Class)

APLRSLIK (Like)

APLRSMGM (Management Class)

APLRSREF (Reference DD Statement)

APLRSESSE (Security Model Profile Name)
APLRSSEG (Security Model, Generic)
This field can have the following values:

- APLGENER: The profile name refers to a generic data-set profile
- APLNOGEN: No generic data-set profile.

APLRSSTO (Storage Class)

OS/400 File Parameters

- APLR4LIB (OS/400 Library Name)
- APLR4FIL (OS/400 File Name)
- APLR4MBR (OS/400 Member Name)
- APLR4MOP (Member Option)
- APLR4OTY (OS/400 File Type)
- APLR4NRC (Initial Number of Records)
- APLR4MRC (Maximum Number of Records)
- APLR4MXM (Maximum Members per File)
- APLR4IRC (Number of Records per Increment)
- APLR4MXI (Maximum Increments)
- APLR4ACC (Access Authority)
- APLR4TXT (File Text Description)
Special Fields for Pre-Transfer, Post-Transfer, and (for MVS) Post-Conversation User-Exit Routines

APLPTRCP (Address of Area That the Transfer Program Traces)
The trace facility allows you to test a pre-transfer, post-transfer, or post-conversation user-exit routine. The transfer program dumps the storage area pointed to by APLPTRCP, in the length specified in APLPTRCL.

When the transfer program calls one of these user-exit routines, this field is set to zero.

APLPTRCL (Length of Area That the Transfer Program Traces)
The trace facility allows you to test a pre-transfer, post-transfer, or post-conversation user-exit routine. The transfer program dumps the storage area pointed to by APLPTRCP in the length specified by APLPTRCL.

When the transfer program calls one of these user-exit routines, this field is set to zero.

APLPTRCL must contain binary zeros or a valid storage area length. After the transfer program takes the trace, it resets APLPTRCL to zero.

APLPDATE (Date when the Transfer Program Calls a User-Exit Routine)
When the transfer program calls a pre-transfer, a post-transfer, or a post-conversation user-exit routine, APLPDATE contains the current date in the format: YY/MM/DD. It is set by the transfer program.

APLPTIME (Time when the Transfer Program Calls a User-Exit Routine)
When the transfer program calls a pre-transfer, a post-transfer, or a post-conversation user-exit routine, APLPTIME contains the current time in the format: HH:MM:SS. It is set by the transfer program.

APLPSMOD (Identification of Request Type)
This field indicates the type of request handler. It can have either of the following values:

- APLRLOC: The exit is running at the requesting location of the transfer
- APLRREM: The exit is running at the responding location of the transfer.

APLPNXRU (Number of Transferred Request Units)
When the transfer program calls a pre-transfer user-exit routine, this field contains zero.

When it calls a post-transfer or a post-conversation user-exit routine, it contains a value as follows:

- **On the sending node**: If the file transfer was successful, this field contains the number of request units sent during the complete file transfer (including restarts). If the file transfer was not successful, it contains zero.

- **On the receiving node**: This field contains the number of received request units (including restarts), regardless of the success of transferring the complete file.

**Note**: Applies to transfers using the LU 0 protocol only.
APLPNXR (Number of Records Transferred)
When the transfer program calls a pre-transfer user-exit routine, this field contains zero.
When it calls a post-transfer or a post-conversation user-exit routine, it contains the number of records or control intervals sent or received during the file transfer (including restarts).

Applies to NetView FTP V2.2 MVS Only

APLBYTNC (Number of Bytes Transferred - Uncompressed)
This field contains the number of data bytes that have been transferred. The bytes are counted before the data is compressed (or after it is decompressed).
In case of a restarted file transfer, the value in this field refers only to the data transferred during the restart. The value of this field is reset to zero for every restart of an interrupted file transfer.
For interrupted transfers, the counters can significantly vary between the sending and the receiving system. In this case, only the receiving system has a reliable byte-count value for the amount of data written to the data set.
The byte count includes the data contents of records and, for variable length records, also the length field of the record. For a transfer of a partitioned data set with record format VB, this count also includes the record descriptor words.
Note: The format of this field is packed decimal.

APLBYTC (Number of Bytes Transferred - Compressed)
This field contains the number of data bytes that have been transferred. The bytes are counted after the data is compressed (or before it is decompressed).
In case of a restarted file transfer, the value in this field refers only to the data transferred during the restart. The value of this field is reset to zero for every restart of an interrupted file transfer.
For interrupted transfers, the counters can significantly vary between the sending and the receiving system. In this case, only the receiving system has a reliable byte-count value for the amount of data written to the data set.
The byte count includes the data contents of records and, for variable length records, also the length field of the record. For a transfer of a partitioned data set with record format VB, this count also includes the record descriptor words.
Note: The format of this field is packed decimal.

End of Applies to NetView FTP V2.2 MVS Only
APLSECV (Security Verified Indicator)
This field indicates that the security parameters passed to the pre-transfer user-exit routine are verified by the exit routine and that the APLxACUI fields can be used to build the security environment, without verifying and using APLxACPW. It is still required to have APLxACPW filled to pass the NetView FTP keyword verification.

APLSECV can have either of the following values:

APLYES
The exit routine has verified the security parameters and APLxACUI is to be used to build the security environment, without the passwords being verified.

APLNO
The exit routine has not verified the security parameters. The transfer program must verify the security parameters.

APLJOB (Job Submission Indicator)
This field indicates whether job submission is to be performed. This field is checked independent of the return code. This means if a transfer is rejected by a user-exit routine, the APLJOB field can be set to APLNO, so that job submission is suspended.

APLJOB can have either of the following values:

APLYES
Job submission to be performed.

APLNO
Job submission suspended.

Fields Used to Modify the Data Areas for Communicating between User-Exit Routines
The local and global data areas are separate areas that are not part of the APL. The following fields can be used to locate these separate areas:

APLLXDPT
Pointer to the address of the local data area.

APLLXDLN
Pointer to the length of the local data area.

APLGDPT
Pointer to the address of the global data area.

APLGDLN
Pointer to the length of the global data area.
Fields Used for PDS Support

This section contains General-use Programming Interface and Associated Guidance information.

| Applies to NetView FTP V2 MVS Only |

**APLXOPT (Transfer Send Option)**
Specifies the type of transfer required. It can have one of the following values:

- **APLXINS (I)**: Insert the specified PDS members
- **APLXUPDT (U)**: Update the specified PDS members
- **APLXREP (R)**: Replace the specified PDS members

**APLAPXPT (Pointer to APX)**
When the transfer program is transferring a PDS, and a pre-transfer user-exit routine is called, this field contains the address of the APX.

| End of Applies to NetView FTP V2 MVS Only |

Fields Used for Dynamic VSAM Cluster Definition

This section contains General-use Programming Interface and Associated Guidance information.

**APLDYRSA (Average Record Size)**

**APLVTYPE (Cluster Type)**
This field specifies the type of VSAM cluster to be transferred. It can have one of the following values:

- **APLESDS**: Entry sequenced data set
- **APLKSDS**: Key sequenced data set
- **APLLDS**: Linear data set (applies to NetView FTP V2 MVS only)
- **APLNBRD**: Relative record data set (applies to NetView FTP V2.2 MVS only)

**APLDYUPL (Length of User Catalog Password)**

**APLDYMPL (Length of Model Password)**

**APLDYKL (Key Length)**
APLDYKO (Key Offset)

Does Not Apply to NetView FTP V2.2 MVS

APLDYUCP (User Catalog Password)

End of Does Not Apply to NetView FTP V2.2 MVS

APLRDYUP (User Catalog Password)

End of Applies to NetView FTP V2.2 MVS Only

APLDYMPW (Model Password)

APLDYME (Model Entry Name)

APLDYDE (Data Component Entry Name)

APLDYIE (Index Component Entry Name)

QSTAT Options for QUERY and MODIFY Commands

This section contains General-use Programming Interface and Associated Guidance information.

APLQSTAT (QSTAT Parameter Options)

Specifies the values for the following functions:

QUERY ADM function

For the QUERY ADM function, this field can have any of the following values:¹⁹

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APLQWT</td>
<td>WAITING</td>
</tr>
<tr>
<td>APLQWTD</td>
<td>WAITING + DSN</td>
</tr>
<tr>
<td>APLQACT</td>
<td>ACTIVE</td>
</tr>
<tr>
<td>APLQACTD</td>
<td>ACTIVE + DSN</td>
</tr>
<tr>
<td>APLQFIN</td>
<td>FINISHED</td>
</tr>
</tbody>
</table>

¹⁹ In NetView FTP VM Release 1.1 and in NetView FTP VSE Release 1.1 the assembler declarations of the symbols have been changed from EQU to DC.
APLQFIND  FINISHED + DSN
APLQALL   ALL
APLQALLD  ALL + DSN
APLQGEN   GENERAL.

MODIFY function
For the MODIFY function, this field can have one of the following values: 19

APLQHLD   HOLD
APLQRLSE  RELEASE.

Extended Data-Compression Mode Fields
This section contains General-use Programming Interface and Associated Guidance information.

APLCOMPR (Extended Data-Compression Mode)
Specifies the type of compression to be used in the transfer. It can have one of the following values:

APLCPNON   No compression
APLCPSNA   Standard SNA compression or compaction
APLCPADA   Adaptive compression.

Requested Running Mode of Servers
This section contains General-use Programming Interface and Associated Guidance information.

APLSSMOD (Requested Running Mode of Sending Server)
Specifies the running mode for the sending server. It can have one of the following values:

APLCONT   Continuous mode
APLSINGL  Single mode.

APLRSMOD (Requested Running Mode of Receiving Server)
Specifies the running mode for the receiving server. It can have one of the following values:

APLCONT   Continuous mode
APLSINGL  Single mode.
Chapter 13. Accessing the APL

This chapter contains General-use Programming Interface and Associated Guidance Information.

The APL control block can be accessed by macros coded in PL/I, Assembler, C language, or by procedures coded in REXX. These macros and procedures provide mapping of the APL for the following user interfaces:

- Application program interface
- Pre-queueing user-exit routine
- Pre-transfer user-exit routine
- Post-transfer user-exit routine
- Post-conversation user-exit routine.

If you code an application program or user-exit routine in a programming language other than PL/I, Assembler, REXX, or C language, you must define a structure for the APL. The application programs and the user-exit routines must be coded correctly to address the APL:

**PL/I**

The macro can be defined in program storage or can be based on a pointer. The pointer’s name is APLP PTR. It is declared automatically when you invoke DVGAPL with the BASED attribute.

**Assembler**

The DVGAPL macro can be invoked to produce a CSECT or a DSECT structure.

**REXX**

You use the external functions DVGCXRPT and DVGCXRGT.

**C Language**

You use the include member DVGCXCAP.

**DVGAPL Macroinstruction (PL/I)**

Figure 29 shows how to code the DVGAPL macroinstruction in a PL/I application program.

```
%DCL APLSRCE CHARACTER;
%DCL APLSTOR CHARACTER;
%APLSRCE = 'PLI';
%APLSTOR = ' ';
%INCLUDE SYSLIB(DVGAPL);
```

Figure 29. DVGAPL Macroinstruction (PL/I)

**APLSRCE**

When you use DVGAPL in a PL/I program, the value of this variable must always be PLI.

**APLSTOR**

If the value of this parameter is BASED, the DVGAPL macro generates a structure that is based and that has the name DVGAPL BASED(APLP PTR). You must set APLPPTR to the address of the APL. The declaration for APLPPTR is automatically generated in DVGAPL.

---

20 REXX and C language apply to NetView FTP V2 MVS only.
If a value is not specified, or is specified as anything other than BASED, the DVGAPL macro generates a structure that is not based and that has the name DVGAPL.

**DVGAPL Macroinstruction (Assembler)**

Figure 30 shows how to code the DVGAPL macroinstruction in an Assembler application program.

```plaintext
instname DVGAPL DSECT=val
```

*Figure 30. DVGAPL Macroinstruction (Assembler)*

When coding this macroinstruction, replace the value `instname` with the name of the macroinstruction. Replace the value `val` with either **YES** or **NO**. The value **NO** causes the control block to be made a part of the control section where the macro is coded. The control block is given a label that is the same as the name of the macro (that is, DVGAPL). This label can be used to address the control block. The value **YES** causes the control block to be made a dummy control section with the same name as the macro (DVGAPL).

**Coding a REXX Procedure**

Figure 31 shows how to define the variable DVGAPL and how to invoke the external function DVGCXRPT to initialize the APL control block with the prefix (*DVGAPL*).

```plaintext
DVGAPL = D2C(COPIES(/zerodot,APLLEN),APLLEN)  
DVGAPL = DVGCXRPT(DVGAPL,APLINIT,APLID,C)
```

*Figure 31. External Function DVGCXRPT (REXX)*

For more information about application programs written in REXX, refer to Chapter 16, “Using REXX Application Programs and User-Exit Routines” on page 147.

**Coding a C-Language Macroinstruction**

Figure 32 shows how to define the variable and how to invoke the include file DVGCXCAP to initialize the control block.

```plaintext
#include "DVGCXCAP.H"  /* Include APL mapping */
struct DVGAPL *APLPTR;  /* Addressability of APL */
```

*Figure 32. DVGAPL Macroinstruction (C Language)*

For more information on application programs written in C language, refer to Chapter 17, “Using User-Exit Routines Written in C Language” on page 161.
Chapter 14. Using the APX

This chapter applies to NetView FTP V2 MVS only. It contains General-use Programming Interface and Associated Guidance Information.

The Application Program Parameter List Extension (APX) is used only on the requesting system of a file transfer and contains the parameters relating to the transfer of PDS members.

If your application program is written in PL/I or Assembler, mapping for the APX is done by the DVGAPX macro. The file DVGCXAX is also available for use in programs written in C language. If your application program is written in REXX, mapping for the APX is done by REXX-written procedures. You can find a description of how to invoke the macro in Chapter 15, “Accessing the APX” on page 145. If you code a user-exit routine in a programming language other than PL/I, Assembler, C Language, or REXX, you must define the structure of the APX.

The transfer program invokes a pre-transfer user-exit routine before the actual transfer of data. If the transfer involves multiple members of a PDS, the transfer program passes the values of the parameters relating to the PDS members to the exit routine (sending system only) in the form of an APX. The pre-transfer user-exit routine can change certain parameter values of the APX. For a more detailed description of the function of the pre-transfer user-exit routine see Chapter 4, “Writing a Pre-Transfer User-Exit Routine” on page 45.

Layout of the APX

The installation tape for your transfer program contains a mapping macro, DVGAPX, for the APX. Use this module as a reference to the data format of the parameter values.

Control Block Header

DVGAPX (Application Program Parameter List Extension)

APX (Structure Name)

APXID (Control Block Identifier)

APXID must always contain the character string “DVGAPX” when communicating with the transfer program. This field serves the following main functions:

- Enables the transfer program to identify the APX.
- Shows the user-exit routines that the transfer program has correctly initialized the APX.
- Makes it easier to locate the APX in a dump.
If you are communicating with the transfer program through the application program interface, you must initialize the APX parameters to their proper values. As part of this process you must also set the APX identifier as one of the parameter values.

If you are using a pre-transfer user-exit routine, the transfer program provides a properly initialized APX.

**APXTYP (APX Type Select or Exclude)**
Specifies whether the transfer is to involve a selection list or an exclusion list. It can have one of the following values:

- **APXTYPS** The transfer program is to transfer the members of the sending system PDS that are specified in the selection list.
- **APXTYPX** The transfer program is to transfer all members of the sending system PDS, except the ones specified in the exclusion list.

**APXENT (Number of APX Entries)**
Specifies the number of entries specified in the selection or exclusion list.

**APXLEN (Length of APX)**
Specifies the length of the APX.

This field must be modified if the pre-transfer user-exit routine changes the number of members selected or excluded.

---

### Fields for Selecting a Member

**APXSNME (Selected Member Name)**
Specifies the name of a PDS member on the sending system that is to be transferred.

**APXRNME (Newname for Member)**
Specifies the name of the transferred PDS member on the receiving system; if the name is to be the same as on the sending system, this field can be left blank.

**APXOPT (Transfer Option on Member Level)**
Specifies how the receiving system is to apply the transferred member. It can have any of the following values:

- **APXOPTI** Insert member in the receiving PDS
- **APXOPTR** Replace the existing receiving PDS member
- **APXOPTU** Update the existing receiving PDS member.

---

### Fields for Excluding a Member

**APXXNME (Excluded Member Name)**
Exclude member from the list of sending system PDS members to be transferred.
Chapter 15. Accessing the APX

This chapter applies to NetView FTP V2 MVS only. It contains General-use Programming Interface and Associated Guidance Information.

The APX control block can be accessed by macros coded in PL/I, Assembler, C language, or by procedures coded in REXX.

If you code an application program or a user-exit routine in a programming language other than PL/I, Assembler, C Language, or REXX, you must define a structure for the APX. The application program and the pre-transfer user-exit routine must ensure that the APX is addressable.

**PL/I**
The macro can be defined in program storage or can be based on a pointer. The pointer’s name is APXPPTTR. It is declared automatically when you invoke DVGAPX with the BASED attribute.

**Assembler**
The DVGAPX macro can be invoked to produce a CSECT or a DSECT structure.

**REXX**
You use the procedures DVGCXRGT and DVGCXRPRT.

**C Language**
You use the include member DVGCXCAX.

**DVGAPX Macroinstruction (PL/I)**
Figure 33 shows how to code the DVGAPX macroinstruction in a PL/I application program.

```plaintext
%DCL APXSRC CHARACTER;
%DCL APXSTOR CHARACTER;
%APXSRC = 'PLI';
%APXSTOR = ' ' | 'BASED';
%INCLUDE SYSLIB(DVGAPX);
```

*Figure 33. DVGAPX Macroinstruction (PL/I)*

**APXSRC**
When you use DVGAPX in a PL/I program, the value of this variable must always be PLI.

**APXSTOR**
If the value of this parameter is BASED, the DVGAPX macro generates a structure that is based and that has the name DVGAPX BASED(APXPPTTR). You must set APXPPTTR to the address of the APX. The declaration for APXPPTTR is automatically generated in DVGAPX.

If a value is not specified, or is specified as anything other than BASED, the DVGAPX macro generates a structure that is not based and that has the name DVGAPX.
**DVGAPX Macroinstruction (Assembler)**

Figure 34 shows how to code the DVGAPX macroinstruction in an Assembler application program.

```
instname DVGAPX DSECT=val
```

*Figure 34. DVGAPX Macroinstruction (Assembler)*

When coding this macroinstruction, replace the value `instname` with the name of the macroinstruction.

Replace the value `val` with either **YES** or **NO**. The value **NO** causes the control block to be made a part of the control section where the macro is coded. The control block is given a label that is the same as the name of the macro (that is, DVGAPX). This label can be used to address the control block. The value **YES** causes the control block to be made a dummy control section with the same name as the macro (DVGAPX).

**Coding a REXX Procedure**

Figure 35 shows how to define the variable DVGAPX, and how to code the external function DVGCXRPT to initialize the APX control block with the prefix (*DVGAPX*).

```
DVGAPX = D2C(COPIES(/zerodot,APXLEN),APXLEN)
DVGAPX = DVGCXRPT(DVGAPX,APXACCR,APXID,C)
```

*Figure 35. External Function DVGCXRPT (REXX)*

For more information about application programs written in REXX, refer to Chapter 16, “Using REXX Application Programs and User-Exit Routines” on page 147.

**Coding a C-Language Macroinstruction**

Figure 36 shows how to define the variable and how to invoke the include file DVGCXCAP to initialize the control block.

```
#include "DVGCXCAH.H"  /* Include APX mapping */
struct DVGAPX *APXPTR;  /* Addressability of APX */
```

*Figure 36. DVGAPX Macroinstruction (C Language)*

For more information on application programs written in C language, refer to Chapter 17, “Using User-Exit Routines Written in C Language” on page 161.
Chapter 16. Using REXX Application Programs and User-Exit Routines

This chapter applies to NetView FTP V2 MVS only. This chapter describes:

- How to call NetView FTP V2 MVS from an application program written in REXX.
- How NetView FTP V2 MVS calls a user-exit routine written in REXX.

REXX Naming Conventions

The following describes the REXX naming conventions used for the control blocks and the fields in the control blocks.

Control Blocks

In a REXX procedure, the application control block, the message area, the query response area, and the application control block extension are variables. These variables must have the following names:

- **DVGAPL** Application control block
- **DVGMSG** Message area
- **DVGQRA** Query response area
- **DVGAPX** Application control block extension.

These names must be used because they are also used by the interface load modules supplied with NetView FTP V2 MVS. The interface load modules are:

- **DVGCXRRIA**
- **DVGCXRIX**
- **DVGCXFOL**.

Fields in the Control Blocks

The field names that already exist in the APL must be used when the REXX programs access fields in a control block. For example, use the field names APLRC, APLBC, and APLLNGTH as arguments for the external functions DVGCXRGT and DVGCXRPT to access the fields in the APL control block.

Similarly, the field names that already exist in the APX must be used. For example, use the field names APXID and APXTYP.

Constants or Keywords

The first command in a REXX procedure must be the call command of the internal subroutines. This command must be in the form **CALL DVGCXD1, CALL DVGCXD2, CALL DVGCXD3, and CALL DVGCXD4**. These subroutines define the keywords and constants that are needed to name the fields of the control blocks. The subroutines are coded in the sample program DVGCXRA1.
Defining the Control Blocks

DVGAPL Variable
In the subroutine DVGCXD1, the field APLLEN defines the length of the APL control block. The REXX function COPIES defines a string that is the length of the DVGAPL, and fills the string with decimal zeros. The REXX function D2C converts the initial value into binary zero. This initializes the APL to X’0’. The following example shows how to define the DVGAPL.

DVGAPL = D2C (COPIES(0,APLLEN),APLLEN)

DVGMSG Variable
The following example shows how to define the message area.

DVGMSG = COPIES(‘ ’,200)
In this example the message area is 200 bytes long and is erased to blank.

DVGQRA Variable
The following example shows how to define the query response area.

DVGQRA = COPIES(‘ ’,500)
In this example the query response area is 500 bytes long and is erased to blank.

DVGAPX Variable
The following example shows how to define the application program extension.

DVGAPX = D2C(COPIES(0,500),500)
In this example the application program extension is 500 bytes long and is erased to X’0’.

User-Exit Variables
The transfer program uses the following variables to send data to the user-exit routines or to post-transfer programs:

DVGSPDAT
Supplies data to the post-transfer program at the sending system.

DVGRPDAT
Supplies data to the post-transfer program at the receiving system.

DVGUXDAT
Supplies data to user-exit routines.

Refer to the NetView FTP V2 MVS User’s Guide for more information on these variables.
Using the External Functions

The application program uses the REXX external functions because it is not possible to structurally define variables in a REXX procedure. These external functions are:

- DVGCXRPT (written in REXX)
- DVGCXRGRT (written in REXX)
- DVGCXFOF (written in Assembler and delivered as a load module).

The application program uses these external functions to send a parameter and its value to the correct position in the APL control block or in the APX control block. The external functions also define the correct length and format of the parameter in the control blocks.

External Function DVGCXRPT

You use the external function DVGCXRPT to supply the APL control block and the APX control blocks with values. Figure 37 shows how to code the statement.

```
DVGAPL = DVGCXRPT(PARM-1,PARM-2,PARM-3,PARM-4,PARM-5)
DVGAPX = DVGCXRPT(PARM-1,PARM-2,PARM-3,PARM-4,PARM-5)
```

*Figure 37. DVGCXRPT External Function*

The values the parameters can have are explained in the following.

**PARM-1**
- Defines the APL control block name (DVGAPL) or APX control block name (DVGAPX).

**PARM-2**
- Defines the value of the field specified by PARM-3.

**PARM-3**
- Defines the APL or APX control-block field names.

This parameter defines the field name of the APL or APX control block, such that DVGCXRPT sends the value of PARM-2 to the correct position in the control block.

**PARM-4**
- Defines the data type of the field specified by PARM-3.

Use either the constant N for a numeric field, the constant C for a character field, or the constant A for an address field. Do not use other characters. Specifying these characters does the following:

- **C**: Field values sent to the control blocks are left-justified. The rest of the field (to the right of the first value) fills with blanks up to the length of the APL or APX field.

- **N**: Field values sent to the control blocks are right-justified. The rest of the field (to the left of the first value) fills with zeros up to the length of the APL or APX field. The values are then converted from decimal into binary.

- **A**: Address field.
PARM-5
Defines the index for the APX select member or exclude member field.

This parameter is optional. It is used when data for select member or exclude member must be filled in the APX. The value of PARM-5 must be numeric. The index shows if the 1st, 2nd, 3rd, ... nth member is filled.

Note: The field APXTYP must be filled with the correct member type; s for select member or x for exclude member.

Examples
The following examples show how to code the external function DVGCXRPT:

- To initialize the APL control block with the prefix (*DVGAPL*), use the following:
  \[DVGAPL = DVGCXRPT(DVGAPL, APLINIT, APLID, C)\]

- To set the length of the APL, use the following:
  \[DVGAPL = DVGCXRPT(DVGAPL, LENGTH(DVGAPL), APLLENGTH, N)\]

- To set the version identification of the APL control block, use the following:
  \[DVGAPL = DVGCXRPT(DVGAPL, APLVIDC, APLVID, C)\]

- To set the constants, use the following:
  \[DVGAPL = DVGCXRPT(DVGAPL, 'XXEASV6', APLRLUNM, C)\]

- To initialize APX control block with the prefix (*DVGAPX*), use the following:
  \[DVGAPX = DVGCXRPT(DVGAPX, APXACCR, APXID, C)\]

- To set the length of the APX, use the following:
  \[DVGAPX = DVGCXRPT(DVGAPX, 1000, APXLEN, N)\]

- To set the number of entries in the APX, use the following:
  \[DVGAPX = DVGCXRPT(DVGAPX, 3, APXENT, N)\]

- To set the member type to s in the APX, use the following:
  \[DVGAPX = DVGCXRPT(DVGAPX, APXTYPS, APXTYP, C)\]

- To set the name of the first select member in the APX, use the following:
  \[DVGAPX = DVGCXRPT(DVGAPX, 'XXXXXX', APXSNME, C, 1)\]

- To set the new name of the first replacement member in the APX, use the following:
  \[DVGAPX = DVGCXRPT(DVGAPX, 'YYYYYY', APXRNME, C, 1)\]

- To set the insert option in the APX, use the following:
  \[DVGAPX = DVGCXRPT(DVGAPX, APXOPTI, APXOPT, C, 1)\]

Error Handling
When invoking the external function DVGCXRPT, you must assign at least four parameters. The fifth parameter is optional. The number of parameters is checked by the REXX function ARG. If less than four parameters are present, the function terminates with the following error message, together with return code 4:

'INCORRECT FUNCTION CALL DVGCXRPT' 'DVGCXRPT ERROR EXIT'
If the external function does not include the characters C, N, or A in the fourth parameter, the function terminates with the following error message, together with return code 8:

'TYP INDICATOR NOT -C- OR -N- OR -A- ' = TYP

If the 5th parameter is specified, and it is nonnumeric, the external function terminates with the following error message, together with return code 4:

'INCORRECT ARGUMENT FIVE IN DVGCXRPT' 'DVGCXRPT ERROR EXIT'

Character Fields and Numeric Fields

If you specify C (character) for PARM-4, the initial value of PARM-2 is filled with blanks to the right of the value. This is done by the REXX function LEFT. The length parameter for the LEFT function is determined using external function DVGCXFOL. This external function is described in “External Function DVGCXFOL (Assembler)” on page 153. After determining the length, the string is put to the APL control block or APX control block using the REXX function OVERLAY. The position parameter for the OVERLAY function is also determined by the external function DVGCXFOL. An example of how to code the statement is shown in the following:

field = OVERLAY(LEFT(INIT,DVGCXFOL(FIELD,L),APL,DVGCXFOL(FIELD,O))

where field is the name of the changed field.

If you specify N (numeric) for PARM-4, the initial value from PARM-2 is filled with zeros to the left of the value. This is done with the REXX function RIGHT. If the length of the RIGHT function is smaller than 4, it is set to 4 (fullword-length). Otherwise, the length of the REXX variable is filled with zeros. This length is then converted from decimal to binary by the REXX function D2C. The length parameter for the D2C function is determined by the function DVGCXFOL. After determining the length, the string is put to the APL control block or APX control block using the REXX function OVERLAY. The position parameter for the OVERLAY function is also determined by the external function DVGCXFOL. An example of how to code the statement is shown in the following:

ZLENGTH = LENGTH(INIT)  
IF ZLENGTH < 4 THEN ZLENGTH = 4  
ZINIT = RIGHT(ZINIT,ZLENGTH,/)zerodot/  
field = OVERLAY(D2C(ZINIT,DVGCXFOL(FIELD,L)),APL,DVGCXFOL(FIELD,O))

where field is the name of the changed field.

The external function DVGCXRPT now returns the changed control block.

External Function DVGCXRGT

You use the external function DVGCXRGT to get values from the APL and APX. All numeric fields in the APL control block or the APX control block are in binary format. Therefore, it is not possible in REXX to process the fields directly. Additionally, because there are no structures for the control blocks, each field of the control block must be converted and loaded into a new REXX variable using the function DVGCXRGT. Figure 38 shows how to code the function.

```plaintext
X = DVGCXRGT(PARM-1,PARM-2,PARM-3,PARM-4)
```

Figure 38. DVGCXRPT External Function
The values the parameters can have are explained in the following.

**PARM-1**
Defines the APL control block name (DVGAPL) or the APX control block name (DVGAPX).

**PARM-2**
Defines the APL or APX control-block field names. Use this parameter to specify the fields from where data should be extracted.

**PARM-3**
Data type of the control-block field name. Use one of the following constants:

- **N** Numeric field. When you specify this constant, the values from the control block are converted from binary into decimal and made available.
- **C** Character field. When you specify this constant, the values from the control-block fields remain unchanged.
- **A** Address. When you specify this constant, an address is accessed in the control block and converted into a hexadecimal address with the REXX function `C2X`. The converted address is then made available.

**PARM-4**
Defines the member index in the APX for the select member or exclude member table entry.

This parameter is optional and is used when data for a select member or exclude member is to be accessed from the APX. The value of PARM-4 is numeric. The index shows if the 1st, 2nd, 3rd ... nth table entry for a member is accessed.

**Example**
The following examples show how to code function DVGCXRGT:

- To access the numeric (binary) fields in the APL, use the following:
  
  \[
  X = DVGCXRGT(DVGAPL,APLLNGTH,N) \text{ or } X = DVGCXRGT(DVGAPL,APLRC,N)
  \]

- To access the character fields in the APL, use the following:
  
  \[
  X = DVGCXRGT(DVGAPL,APLRLUNM,C)
  \]

- To access the message-area address in the APL, use the following:
  
  \[
  X = DVGCXRGT(DVGAPL,APLMAPTR,A)
  \]

- To access the name of a select member in the APX, use the following:
  
  \[
  DVGAPX = DVGCXRGT(DVGAPX,APXSNME,C,1)
  \]

**Error Handling**
When coding the function DVGCXRGT, you must assign at least three parameters. The number of parameters is checked by the REXX function `ARG`. If less than three parameters are present, the function terminates with the following error message, together with return code 4:

'INCORRECT FUNCTION CALL DVGCXRGT'
'DVGCXRGT ERROR EXIT'
If you did not specify C, N, or A for PARM-3, the function terminates with the following error message, together with return code 8:

'TYP INDICATOR NOT -C-, -N- or -A- = TYP'
'DVGCRXGT ERROR EXIT'

If you specify PARM-1 to be DVGAPX, you must specify PARM-4. The value of PARM-4 must be numeric. If it is not, the function terminates with the following error message, together with return code 4:

'INCORRECT ARGUMENT FOUR IN DVGCRXGT'
'DVGCRXGT ERROR EXIT'

Character Fields and Numeric Fields

If you specify C (character) for PARM-3, the value from the APL or APX field (PARM-2) is accessed with the REXX function SUBSTR. The length parameter for the SUBSTR function is determined with the REXX function DVGCXFOL. An example of how to code the statement is shown in the following:

\[ X = \text{Substr(Area,DVGCXFOL(FIELD,O),DVGCXFOL(FIELD,L))} \]

If you specify N (numeric) for PARM-3, the value from the APL or APX control block is accessed with the REXX function SUBSTR and converted from binary to decimal with the REXX function C2D. The length and offset parameters are also determined with the function DVGCXFOL. An example of how to code the statement is shown in the following:

\[ X = \text{C2D(SUBSTR(AREA,DVGCXFOL(FIELD,O),DVGCXFOL(FIELD,L))} \]

If you specify A (address) for PARM-3, the value from the APL block is accessed with the REXX function SUBSTR. The value is then converted from character to hexadecimal with the REXX function C2X. The address can then be used for the REXX function STORAGE to access or change data at that address. An example of how to code the statement is shown in the following:

\[ X = \text{C2X(SUBSTR(AREA,DVGCXFOL(FIELD,O),DVGCXFOL(FIELD,L))} \]

If you specify PARM-4, the APX is checked to determine whether s for select or x for exclude is specified. Depending on this, the relevant field (PARM-3) from the APX is determined using the member index (PARM-4), and returned to the invocation position.

External Function DVGCXFOL (Assembler)

The external function DVGCXFOL is written in Assembler language and supplied as a load module. You can use this function to determine the offset of a field or determine the length of a field in the APL control block or APX control block. DVGCXRPT and DVGCXRGT also use this function when they are instructed to access a field of the APL or APX control block. Figure 39 shows how to code the function.

\[ X = \text{DVGCXFOL(PARM-1,PARM-2)} \]

Figure 39. DVGCXFOL External Function

Chapter 16. Using REXX Application Programs and User-Exit Routines 151
The values the parameters can have are explained in the following.

**PARM-1**
Defines an APL field name or an APX field name.

**PARM-2**
Defines whether field offset (O) or field length (L) is used. Specifying these values detects the following:

- **O** Field offset
- **L** Field length.

**Examples**
The following examples show how to code the external function DVGCXFOL.

- To detect the offset of the field APLID in the APL, use the following:
  \[
  X = \text{DVGCXFOL}(\text{APLID}, \text{O})
  \]
- To detect the length of the field APLID in the APL, use the following:
  \[
  X = \text{DVGCXFOL}(\text{APLID}, \text{L})
  \]

**Function Return Value**
The external function returns either the offset or the length of an APL or APX field.

**Error Handling**
Before the external function runs, it is checked for valid parameters. If the statement contains no parameters, return code 8 is set in register 15, and the external function terminates. If PARM-2 is not given, return code 12 is set in register 15, and the external function terminates. In both cases, no data is returned and REXX interprets this as an error.

If the field does not exist, return code 4 is set in register 15, and the external function terminates.

---

**Calling NetView FTP V2 MVS from a REXX Application Program**

You can use the REXX external function DVGCXRIA to call the NetView FTP V2 MVS module DVGAPI.

**Note:** This is similar to a PL/I application program calling the NetView FTP V2 MVS module DVGIFAI.

The REXX external function DVGCXRIA prepares and offers the APL and APX control blocks, message area, and query response area to NetView FTP V2 MVS. It also adds to the APL the addresses of the message area, query response area, and APX, and transfers control to NetView FTP V2 MVS via the DVGAPI.

When NetView FTP V2 MVS returns control to DVGCXRIA, the control blocks are made available to the REXX application program where they can be inspected by the external function DVGCXRGT.
Calling the External Function DVGCXRIA

To call DVGCXRIA, code the statement as shown in Figure 40.

```
CALL DVGCXRIA (PARM-1,PARM-2,PARM-3,PARM-4,PARM-5,PARM-6,PARM-7,PARM-8,PARM-9)
```

Figure 40. DVGCXRIA External Function

The values the parameters can have are explained in the following.

**PARM-1**
Defines the APL control block name (DVGAPL). You must specify a value for this parameter.

**PARM-2**
Defines the message area (DVGMSG). This parameter is optional.

**PARM-3**
Defines the query response area (DVGQRA). This parameter is optional.

**PARM-4**
Defines the APX control block (DVGAPX). This parameter is optional.

**PARM-5**
Defines DVGSPDAT. This parameter is optional.

**PARM-6**
Defines DVGRPDAT. This parameter is optional.

**PARM-7**
Defines DVGUXDAT. This parameter is optional.

**PARM-8**
Defines DVGSLF. This parameter is optional.

**PARM-9**
Defines DVGRLF. This parameter is optional.

For more information about PARM-5, PARM-6, PARM-7, PARM-8, and PARM-9 refer to “What DVGCXRIA Does” on page 156.

Specify these parameters in the sequence shown. If you do not specify the correct sequence, an error occurs and the routine terminates.

The control blocks are assigned to the subroutine in the following sequence:

DVGAPL,DVGMSG,DVGQRA,DVGAPX,DVGSPDAT,DVGRPDAT,DVGUXDAT,DVGSLF,DVGRLF

You must specify PARM-1, the other parameters are optional. If you omit an optional parameter, include a comma (,) at the position of the omitted parameter, unless it is at the end of the sequence.

**Example**
The following example shows how to call the external function DVGCXRIA:

```
X = DVGCXRIA(DVGAPL,DVGMSG,,DVGSPDAT,DVGRPDAT)
```
Return Values
The external function returns the return code from the REXX Variable Access Interface (IRXEXCOM). These codes are described in TSO/E Version 2—REXX/MVS Reference.

When NetView FTP V2 MVS returns control to the external function, the following control blocks are available to the REXX application program:

- DVGAPL
- DVGMSG
- DVGQRA
- DVGAPX
- DVGSPDAT
- DVGRPDPAT
- DVGUXDAT
- DVGSLF
- DVGRLF.

Error Handling
The external function checks the transferred parameters. If no parameters are transferred, return code 4 is set in register 15 and returned to the calling location.

What DVGCXRRIA Does
DVGCXRRIA checks the call statement to determine which parameters are specified and then:

- If PARM-2 is specified, the address of the message area is loaded into the APL field APLMAPTR.
- If PARM-3 is specified, the address of the query response area is loaded into the APL field APLFBAP.
- If PARM-4 is specified, the address of the APX area is loaded into the APL field APLAPXPT.
- If PARM-5 is specified, the address of the data area for post-transfer (sending) program is loaded into the APL field APLSPPTR.
- If PARM-6 is specified, the address of the data area for post-transfer (receiving) program is loaded into the APL field APLRPPTR.
- If PARM-7 is specified, the address of the data area for the user exit is loaded into the APL field APLUXPTR.
- If PARM-8 is specified, the address of the data area for the sending file ID is loaded into the APL field APLSLFA.
- If PARM-9 is specified, the address of the data area for the receiving file ID is loaded into the APL field APLRLFA.

Next, DVGCXRRIA calls the NetView FTP V2 MVS module DVGIFAI and transfers the address of the APL control block to NetView FTP V2 MVS. When the subroutine is returned from NetView FTP V2 MVS, the following REXX variables are set with the REXX interface VARIABLE ACCESS (IRXEXCOM):

- DVGAPL
- DVGMSG
- DVGQRA
- DVGAPX
- DVGSPDAT
- DVGRPDPAT
- DVGUXDAT
- DVGSLF
- DVGRLF

Only the control blocks specified in the function call are made available to the REXXX application program. A control-block field can then be inspected by using the external function DVGCXRGT.
Invoking a User-Exit Routine from NetView FTP V2 MVS

The following assembler-written modules are supplied for calling a user-exit routine from NetView FTP V2 MVS:

- DVG CXRIX
  This is the base interface module for invoking a user-written REXX procedure. It is supplied as a load module.

- DVG CXRIU
  This is the user interface module for parameter definition:
  
  **EXITNAME**  
  Name of the user-written REXX procedure

  **EXDDNAME**  
  DD name of a PO data set containing the user-written REXX procedure.

What DVG CXRIX Does

NetView FTP V2 MVS uses DVG CXRIX to invoke a user-exit routine written in REXX. DVG CXRIX does the following:

- Prepares the parameter list and REXX interface blocks
- Initiates a REXX environment
- Passes control to the user-written REXX procedure
- On return, passes control of the APL and APX control blocks to NetView FTP V2 MVS
- Returns control to NetView FTP V2 MVS.

DVG CXRIX calls the REXX routine IRXEXEC. To invoke IRXEXEC, the following parameters are used:

**EXECBLOCK**
This parameter specifies the name of the REXX procedure you want to invoke, and the DD name of the data set. The default value for the REXX procedure is DVG CXRU1, and SYSEXEC for the DD name.

**Argument List**
Only the address and length of the APL are passed as arguments from NetView FTP V2 MVS to IRXEXEC.

**FLAG**
This parameter tells the routine IRXEXEC that the APL will be returned from the REXX procedure.

**EVALBLOCK**
This parameter tells IRXEXEC to expect the APL back from the REXX program. The APL is then reloaded by NetView FTP V2 MVS.

Defining the Name of the User-Exit Routine

The module DVG CXRIX needs the name of the user-written REXX procedure and the DD name of the data set containing the REXX EXEC. Specify these parameters in the assembler-written module DVG CXRIU. This module is then assembled and linked together with DVG CXRIX. The entry point of the resulting load module must be DVG CXRIX. The resulting module can have any name. Because the sample user-exit routine has the name DVG CXRU1, this name is used as a default in DVG CXRIU. This name must be used for the server initialization parameters PPEXIT, PRETRAN, POSTTRAN, or POSTCONV.
The library containing the REXX EXEC must be allocated to the NetView FTP V2 MVS server job, using the DD name specified in module DVGCXRIU. The default DD name is SYSEXEC.

**What the REXX User-Exit Routine Can Do**

The REXX interface DVGCXRIX passes the APL control block as one REXX variable DVGAPL to the REXX EXEC. The EXEC must receive the APL by coding PARSE ARG DVGAPL. The EXEC code may inspect and modify individual fields in the APL using the external functions DVGCXRGT and DVGCXRPT, described previously. When the EXEC finishes, it must return the DVGAPL to the interface DVGCXRIX (RETURN DVGAPL).

**How to Link the User-Exit Routine**

DVGCXRIX must be linked with the user-exit routine and must be the entry point of the user-exit routine load module. The appropriate link-edit control statements are shown in Figure 41.

```
INCLUDE NetView FTP V2 MVS-load-library(DVGCXRIX)
ENTRY DVGCXRIX
NAME user-exit routine name(R)
```

*Figure 41. DVGCXRIX Link-Edit Control Statements*

You are recommended to use the same name for the user-exit routine and the REXX EXEC.

**Positioning of Values in the Message Area**

When NetView FTP V2 MVS calls a REXX user exit, the address of the message area in the APL control block is the APLMAPTR field. To supply the message area with values, the REXX function `STORAGE` is required. To simplify the processing of the message area, the REXX external function `DVGCXRMP` is provided.

Before calling DVGCXRMP, the address of the message area must be accessed from the APL. This is done with the external function DVGCXRGT.

The following example shows how to get the message area address:

```
MSGADR = DVGCXRGT(DVGAPL,APLMAPTR,A)
```

- **MSGADR**
  - This is a REXX variable.
- **DVGAPL**
  - Defines the APL control block name.
- **APLMAPTR**
  - Defines the address in the APL.
- **A**
  - Specifies that the address is converted into hexadecimal numbers.
External Function DVGCXRMP (REXX)

You use the external function DVGCXRMP to supply the message area with values. The following example shows how to code the statement.

```rexx
MSGADR = DVGCXRMP(PARM-1,PARM-2,PARM-3)
```

**PARM-1**
Defines the address of the message area where a value is to be loaded.

**PARM-2**
Defines the value to be loaded in the form of constants or variables.

**PARM-3**
Defines the data type of the fields in the message area.

Use either the constant N for a numeric, or the constant C for a character field.
Do not use other characters. Specifying these characters does the following:

- **C** Field values set in the message area are left-justified.
- **N** Field values set in the message area are right-justified. The rest of the field (to the left of the first value) fills with zeros. The values are then converted from decimal into binary.

**Examples**
Depending on the data type specified in PARM-3, the function returns the new address to where the next value should be loaded. Therefore, the message area should always be loaded from left to right.

The following examples show how to code the external function DVGCXRMP:

- To define the number of messages in the message area, use the following:
  ```rexx
  MSGADR = DVGCXRMP(MSGADR,1,N)
  ```
  In this example, the MSGADR value before invocation is X'6400' and after invocation is X'6404'.

- To define a message length, use the following:
  ```rexx
  MSGADR = DVGCXRMP(MSGADR,length(MSGCSS) + 4,N)
  ```
  In this example, the MSGADR value after invocation is X'6408'.

- To define a message, use the following:
  ```rexx
  MSGADR = DVGCXRMP(MSGADR,MSGCSS,C)
  ```
  In this example, the MSGADR value after invocation is X'6418'.

**Error Handling**
When coding the external function DVGCXRMP, you must assign at least three parameters. The number of parameters is checked by the REXX function ARG. If fewer than three parameters are present, the function terminates with the following error message, together with return code 4:

```
"INCORRECT FUNCTION CALL DVGCXRMP"
"DVGCXRMP ERROR EXIT"
```

If you do not specify either C or N for PARM-3, the function terminates with the following error message, together with return code 8:

```
"TYP INDICATOR NOT -C- OR -N- = TYP"
```
Special Considerations when Processing the APX Control Block

If an APX control block is processed by the exit routine, it must be available as a REXX variable. The address of the APX is in the APL control block. First check if there is an address in field APLAPXPT (pointer for APX). The APX can be processed with the functions DVGCXRGT and DVGCXRPT. After the processing, the complete APX must be loaded back into the address of the APL.

To process the APX, two REXX EXECs are provided. DVGCXRAV returns the APX as a REXX variable. DVGCXRVA stores the APX passed to it as a variable at the address contained in the APL field APLAPXPT. To receive the APX as a variable, code the statement as shown in the following:

\[
\text{DVGCAPX} = \text{DVGCXRAV(DVGAPL)}
\]

When the address field APLAPXPT does not contain an address, then DVGCXRAV returns a null string.

Fields of the DVGCAPX can be inspected or modified using DVGCXRGT and DVGCXRPT.

To store the modified DVGCAPX at the address contained in the APL field APLAPXPT, code as shown in the following:

\[
\text{MYRC} = \text{DVGCXRVA(DVGAPL, DVGCAPX)}
\]

DVGCXRVA returns the following return codes:

- **0**: The APX has been stored.
- **4**: The APX has been stored — the original APX was larger.
- **8**: The original APX was smaller. This is an error condition.
- **12**: The DVGAPL does not contain an APX address. This is an error condition.
Chapter 17. Using User-Exit Routines Written in C Language

This chapter applies to NetView FTP V2 MVS only.

NetView FTP V2 MVS and application programs written in C language cannot communicate directly with each other. An interface, written in Assembler, provides the necessary interface between NetView FTP V2 MVS and the C-written application program. Such an interface routine is called an assembler function or an assembly program. NetView FTP V2 MVS passes the address of the Application Program Parameter List (APL) to the assembly program when it calls a user-exit routine. It is not possible to pass a pointer value to a C-main function. This problem is solved using an assembly-interface program and the C-library routine SSCANF (READ DATA). The name of the assembly-interface program is DVGCXCUI.

Note: For more information about C-library routines refer to SAA Common Programming Interface - C Reference.

What DVGCXCUI Does

The assembly-interface routine DVGCXCUI connects the user-exit routine. DVGCXCUI is the entry point of the C-language load module that is called by NetView FTP V2 MVS. Therefore, DVGCXCUI must be linked with the user-exit routine.

The assembly interface converts the address of the APL into an 8-byte hexadecimal string, sets up the parameter appropriate for a C-main function, and calls the C-exit program via the entry CEESTART.

What the User-Exit Routine Does

To receive the pointer to the APL, the C-main function must specify the parameters ARGC and ARGV, and must obtain the APL address using the C-library function SSCANF (READ DATA). Figure 42 shows a sample prolog of the C-language main function.

#include "DVGCXCAP.H" /* Include APL mapping */

main (int argc, char *argv[])
{
    struct DVGAPL *APLPTR; /* Addressability of APL */
    sscanf(argv[argc-1], "%p", &APLPTR); /* Obtains APL pointer */
    :
    :
}

Figure 42. Sample Prolog of a C-Language Main Function

Note: The C-mappings (structure type definitions) for the NetView FTP V2 MVS interface control blocks APL, APX, QAR, and QSR are provided as include files in the NetView FTP V2 MVS sample library. Their names are DVGCXCAP, DVGCXCAX, DVGCXCQR, and DVGCXCQS.
How to Link the User-Exit Routine

DVGCXCU1 must be linked with the user-exit routine. DVGCXCU1 must be the entry point of the user-exit routine load module. The appropriate link-edit control statements are shown in Figure 43 on page 162.

```
INCLUDE NetView FTP V2 MVS-load-library(DVGCXCU1)
INCLUDE syslib(C-main program name)
ENTRY DVGCXCU1
NAME user-exit routine name(R)
```

Figure 43. DVGCXCU1 Link-Edit Control Statements
Glossary

This glossary defines many of the terms and abbreviations used with NetView FTP. If you do not find the term you are looking for, refer to the Dictionary of Computing, New York: McGraw-Hill, 1994.

Access Method Services. A utility program that defines VSAM data sets and allocates space for them, converts indexed sequential data sets to key-sequenced data sets with indexes, modifies data set attributes in the catalog, facilitates data set portability between operating systems, creates backup copies of data sets and indexes, helps make inaccessible data sets accessible, and lists data set records and catalog entries.

ACF/VTAM. Advanced Communications Function for the Virtual Telecommunications Access Method.

active request. A request that is presently being scheduled or for which the corresponding file transfer is in progress.

adaptive compression. A method by which the amount of storage required for data can be reduced by replacing character strings that are repeated with references to a directory of such character strings.

added request. A request that has been added to the request queue.

administrator query command. A type of command that causes NetView FTP to retrieve information about the request queue or the requests in it; the information retrieved is intended for the NetView FTP administrator. Contrast with user query command.

administrator query record. A record containing the information NetView FTP retrieves when an administrator issues an administrator query command. Contrast with user query record.

adopted authority. When a program is created, it can specify that the program always runs under the program owner’s user profile. A user does not need authority specifically given to him for the objects used by the program, but uses (adopts) the program owner’s authority. The user has authority for the objects used by the program only when he is running the program and other programs called by the program.

Advanced Communications Function for the Virtual Telecommunications Access Method. An IBM licensed program that controls communication and the flow of data in an SNA network. It provides single-domain, multiple-domain, and interconnected network capability. VTAM runs under MVS, VSE, and VM/SP, and supports direct control application programs and subsystems such as NetView FTP and VSE/POWER.

Advanced Function Feature. A set of extra functions that extend the capabilities of the NetView FTP V1 MVS base product.

advanced peer-to-peer networking. Data communications support that routes data in a network between two or more APPC systems that are not directly attached.

advanced program-to-program communication. An implementation of the SNA/SDLC LU 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

AET. Application Entity Title.

AFF. Advanced Function Feature.

alias. An alternative name for a member of a partitioned data set.

AMS. Access Method Services.

APAR. Authorized program analysis report.

APF. Authorized Program Facility.

APL. Application program parameter list.

APPC. Advanced program-to-program communication.

Application Entity Title. The name by which an OSI application (and filestore) can be addressed by remote users. By contrast, the local name of the filestore is the filestore nickname.

application program parameter list. A control block used by application programs to pass parameter values to NetView FTP.

application program parameter list extension. A control block used by application programs to pass parameter values to NetView FTP; a supplement to the application program parameter list.

APPN. Advanced peer-to-peer networking.

APX. Application program parameter list extension.

attended mode. An operating mode of NetView FTP for Workstations that assumes that a user at a workstation expects to be informed about the success of transfers and is available to load devices such as diskettes or tapes. Contrast with unattended mode.

automatic logon retry. NetView FTP’s method for eventually establishing a conversation with a remote
system. It is used when a server cannot initiate a conversation with a server at a remote system because:

- All the servers at the remote system are busy with other transfers
- None of the servers at the remote system has been started
- ACF/VTAM is temporarily unable to find a path between the two servers.

The server at the local system automatically changes the status of the request from active back to waiting, and goes on to process the next request in the request queue (MVS, VSE, or VM) or request database (OS/400). Later, NetView FTP tries again to process the request. It keeps trying until it succeeds in initiating a conversation.

**automatic transfer restart.** NetView FTP’s method for automatically restarting a file transfer that was interrupted. In the following situations, NetView FTP is not able to recover a conversation:

- The queue handler or server at either system is terminated
- The server at either system cannot allocate the file being transferred
- A pre-transfer user-exit routine at either system rejected the file transfer
- There is a severe or prolonged conversation outage.

However, when you create a request you can specify that if one of these situations arises, NetView FTP is to change the status of the request back to waiting. The servers at your system then periodically reprocess the request until one of them succeeds in restarting the transfer.

**basic partitioned access method.** An access method that can be used to create program libraries in direct-access storage for convenient storage and retrieval of programs.

**basic sequential access method.** An access method for storing or retrieving data blocks in a continuous sequence, using either a sequential access or a direct access storage device.

**BIND.** Bind Session.

**Bind Session.** In SNA, a request to activate a session between two logical units.

**blocked request.** A waiting request that is trapped in the queue. The request was passed to the queue handler at a time when there was enough CSA storage available to NetView FTP to do so. However, in the meantime, the limit of the amount of CSA storage available to NetView FTP was reduced and is not enough to allow the queue handler to pass the request to a server for processing. The request cannot be processed until the limit of the amount of CSA storage available to NetView FTP is raised.

**BPAM.** Basic partitioned access method.

**BSAM.** Basic sequential access method.

**callback.** In the AIX operating system, a procedure that is called if and when certain specified conditions are met. This is accomplished by specifying the procedure in a callback list.

**callback list.** (1) A list of procedures that are called if and when certain specified conditions are met. (2) In the program IBM AIXwindows Environment/6000, individual widgets can define callback lists.

**CCB.** Command control block.

**CCI.** Common control block insertion.

**CCSID.** Coded character set identifier.

**CCW.** Channel command word.

**CDC.** Character data conversion.

**CDRA.** Character Data Representation Architecture.

**CDRM.** Cross-domain resource manager.

**CDRS.** Cross-domain resource.

**character data.** Data in the form of letters, numbers, and special characters, such as punctuation marks, that can be read by the human eye.

**Character Data Representation Architecture.** An IBM architecture that defines a set of identifiers, services, supporting resources, and conventions to achieve a consistent representation, processing, and interchange of graphic character data in SAA environments.

**character literal.** A symbol, quantity, or constant in a source program that is itself data, rather than a reference to data.

**checkpoint.** A point at which information about the status of a file transfer is recorded. If the file transfer is interrupted, NetView FTP can use this information to resume the file transfer from a point near where the interruption occurred instead of from the beginning of the file.

**checkpoint data set.** A data set that contains information about the current status of an active file transfer. If the file transfer is interrupted, NetView FTP can use this information to resume the file transfer from a point near
where the interruption occurred instead of from the beginning of the file.

**checkpoint file.** A synonym for checkpoint data set.

**checkpoint record.** A record of a checkpoint data set or file. One checkpoint record contains the information needed to restart one file transfer from a checkpoint.

**checkpoint/restart data set.** A deprecated term for checkpoint data set.

**checkpoint/restart record.** A deprecated term for checkpoint record.

**class.** See server class.

**client.** On a local area network, a workstation that requests service from a server workstation.

**CLIST.** Command list.

**Coded character set identifier.** In NetView FTP, an identifier that represents a set of graphic characters and their code point assignment. The coded character set identifier defines how characters are mapped to decimal values.

**command control block.** A control block that contains details of the queue handler command to be carried out.

**command list.** A list of commands and control statements that is assigned a name. When the name is invoked (as a command) the commands in the list are executed.

**common service area.** In MVS, a part of the common area that contains data areas that are addressable by all address spaces. During use, these areas are protected by the key of the requester.

**completion user exit.** Deprecated term for post-transfer user exit.

**compression.** A technique for converting data into a form that requires less storage space and less transmission time than its original form. Contrast with decompression. See also SNA compaction and SNA compression.

**condition code.** A 4-digit decimal value derived from the value a server places in register 15 before returning control to the operating system. The digits of the condition code consist of the server return code and, for a server running in single mode, the file-transfer return code.

**continuous mode.** A server running mode in which a server continues running after it has transferred a file.

**control language.** The set of all commands with which a user requests system functions.

**control point.** A component of an APPN or LEN node that manages the resources of that node. In an APPN node, the CP is capable of engaging in CP-CP sessions with other APPN nodes. In an APPN network node, the CP also provides services to adjacent end nodes in the APPN network.

**control statement.** A statement that controls or affects the running of a program. For example, NOTAFTER=('21:34', '94/12/25') is a control statement that assigns the value ('21:34', '94/12/25') to the parameter represented by the keyword NOTAFTER.

**conversation.** In SNA, a connection between two transaction programs over an LU-LU session that lets them communicate with each other while processing a transaction.

**CP.** Control point.

**CP-CP sessions.** The parallel sessions between two control points, using LU 6.2 protocols.

**Cross-domain resource.** In VTAM programs, synonym for other-domain resource.

**Cross-domain resource manager.** The functions of the system services control point (SSCP) that control initiation and termination of cross-domain resources.

**cryptography key.** A value used to encrypt and decrypt data transmitted in an LU-LU session that uses cryptography.

**cryptography.** The transformation of data to conceal its meaning.

**CSA.** Common service area.

**current request.** The request currently being created or changed by the user of the NetView FTP panels.

**daemon.** In the AIX operating system, a program that runs unattended to perform a standard service.

**daemon process.** In the AIX operating system, a process begun by the root user or the root shell that can be stopped only by the root user. Daemon processes generally provide services that must be available at all times.

**DASD.** Direct-access storage device.

**data control block parameters.** The following parameters: Record Format, Logical Record Length, and Physical Block Size.
**data set control block.** A control block containing specifications for data sets that are to be created.

**data transfer message unit.** The message unit used to send the data object.

**DCB parameters.** Data control block parameters.

**decompression.** A technique for converting compressed data back into its original form. Contrast with compression.

**default first qualifier.** Server initialization parameter. Sometimes referred to as the *GID initialization parameter.* The server uses this parameter as the first qualifier when it creates a name for the data set.

**default value.** The value that is assigned to a parameter by a program if no value is specified by a user.

**deferred request.** A waiting request that is temporarily trapped in the queue. CSA storage was obtained for NetView FTP, but not enough for the queue handler to pass the request to a server for processing. The request is processed later, when more CSA storage is available to NetView FTP.

**delay time.** The amount of time a finished request stays in the request queue before rebuilding the request queue causes it to be deleted automatically.

**direct transfer.** Transfer of data from one file to another file without first storing the data in an intermediate file.

**direct-access storage device.** A storage device for which access time is effectively independent of the location of the data being accessed.

**directory file.** In the AIX operating system, a file that contains information the system needs to access all types of files.

**distribution-service component.** A component of NetView FTP VM that handles communication with the queue handler (such as retrieving requests to be processed), and with the network (such as establishing conversations and transferring files). In NetView FTP VM, each server consists of one distribution-service component and up to 32 file-service components.

**distribution-service machine.** With NetView FTP VM, a virtual machine in which a distribution-service component runs.

**DSCB.** Data set control block.

**DTMU.** Data transfer message unit.

**dynamic allocation.** The allocation of a file when it is needed, not in advance. Contrast with *job allocation.*

**encrypt.** To scramble data or convert data to a secret code that masks the meaning of the data to any unauthorized recipient.

**entry sequence.** The order in which records are physically arranged in auxiliary storage.

**entry-sequenced data set.** A data set whose records are loaded without respect to their contents, and whose relative byte addresses cannot change.

**ESDS.** Entry-sequenced data set.

**ESTAE.** Extended specify task abnormal exit.

**exceptional checkpointing.** To take a checkpoint when certain types of errors occur.

**exchange identification.** The ID that is exchanged with the remote physical unit when an attachment is first established.

**exclude members.** To choose those members of a PDS that are not to be transferred. Contrast with *select members.*

**exit.** A point in a program at which control is passed to another program.

**exit routine.** A routine that receives control when a specified event occurs.

**Exit(n) Message Unit.** The message unit used to convey information provided by the Post-Transfer User Exit routine n (where n is 1 or 2) of the sender to the receiver.

**extended specify task abnormal exit.** A macroinstruction that allows a user to intercept a scheduled abnormal termination.

**FAT.** File allocation table.

**FBA.** Feedback area.

**feedback area.** An area of storage containing information related to a queue handler command. For example, an FBA can contain a *request control block,* a *query data area,* or a *server data area.*

**file allocation table.** A table used by DOS and OS/2 to allocate space on a disk for a file and to locate and chain together parts of the file that may be scattered on different sectors so that the file can be used in a random or sequential manner.

**file group.** One or more files that reside on one system. For example, all files that are stored in the same directory or whose file names consist of partly matching character strings are considered a file group.
file pool. A collection of minidisks managed by SFS. It contains user files and directories and associated control information. Many user's files and directories can be contained in a single file pool.

file transfer. The sending and receiving of the contents of a file.

File Transfer Access and Management. A set of programs, such as OSI/File Services, which conforms to FTAM standards to manage and transfer files over an OSI network.

file-service component. A component of NetView FTP VM that handles file access and the taking of checkpoints. In NetView FTP VM, each server consists of one distribution-service component and up to 32 file-service components.

file-transfer completion message. A message, sent by a server to a user after a file transfer, which describes the outcome of a file transfer.

file-transfer report. A file, sent by a server to a user after a file transfer which describes the outcome of a file transfer.

file-transfer request. A list of parameters and their values that tell NetView FTP (1) that it is to transfer a file from one system to another, and (2) about the file transfer and the sending and receiving data sets.

filestore. See local filestore and remote filestore.

filestore nickname. The name of the filestore at the local level. It is defined by the filestore owner when registering the filestore in the LRD, and is used by authorized local users to access that filestore.

filestore owner. The single user, local or remote, who has created the filestore and who controls the passwords for accessing it and the filestore accessibility. Each OSI/File Services user owns one filestore.

filestore subset. A subdivision of the local filestore. It is the first qualifier of the MVS data-set name. Each local file is registered in the LRD under a related filestore subset, which in turn belongs to a local filestore.

filter. In the AIX operating system, a command that reads standard input data, modifies the data, and sends it to the display screen.

finished request. A request for which the corresponding file transfer has finished, whether successfully or unsuccessfully.

FIU. File Interchange Unit.

FTP level. A character that represents the level of sophistication of an FTP or NetView FTP program.


FSB. NetView FTP shared block.

FSBX. NetView FTP shared block extension.

GDG. Generation data group.

generation data group. A collection of data sets kept in chronological order; each data set is a generation data set.

generation data set. One generation of a generation data group.

GETVIS area. Storage space within a partition or the shared virtual area, available for dynamic allocation to programs.

GiD initialization parameter. See default first qualifier.

GUI. The graphical user interface of NetView FTP for Workstations.

handle. (1) In the Advanced DOS and OS/2 operating systems, a binary value created by the system that identifies a drive, directory, and a file so that the file can be found and opened. (2) In the AIX operating system, a data structure that is a temporary local identifier for an object.

HDAM. Hierarchic direct access method.

hex. Abbreviation of hexadecimal.

hierarchic direct access method. A database access method that uses algorithmic addressability of records in a hierarchic direct organization.

hierarchic indexed sequential access method. A database access method that uses indexed access to records in a hierarchic sequential organization.

High Performance File System. A file organization available under OS/2.

HISAM. Hierarchic Indexed Sequential Access Method.

HPFS. High Performance File System.

ICCF. Interactive computing and control facility.

ICF. Intersystem communications function.

IMS/VS. Information Management System/Virtual Storage.
**independent LU.** A logical unit (LU) that does not receive an ACTLU over a link. Such LUs can act as primary logical units (PLUs) or secondary logical units (SLUs) and can have one or more LU-LU sessions at a time.

**INI file.** See *initialization file*.

**initialization file.** A file that contains parameters that determine how NetView FTP for Workstations starts running.

**input field.** An area on a panel in which data is entered.

**instance.** In the AIX operating system, the concrete realization of an abstract object class. An instance of a widget or gadget is a specific data structure that contains detailed appearance and behavioral information that is used to generate a specific graphical object on-screen at run time.

**Internet.** A wide area network connecting thousands of disparate networks in industry, education, government, and research. The Internet network uses /IP as the standard for transmitting information.

**Internet Protocol.** A protocol used to route data from its source to its destination in an Internet environment.

**Intersystem communications function.** Communications between application programs on an AS/400 system and an application program on a remote system are accomplished using the AS/400 system intersystem communications function (ICF) and the underlying support.

**IP.** Internet Protocol.

**ISPF.** Interactive System Productivity Facility.

**JCL.** Job control language.

**JES.** Job entry subsystem.

**job allocation.** The allocation of a file by a server startup job. The allocation takes place when a server is started, which is before (sometimes long before) the file transfer takes place. Contrast with *dynamic allocation*.

**job control language.** A control language used to identify a job to an operating system and to describe the job's requirements.

**key sequence.** In VSAM, the collating sequence of data records as determined by the value of the key field in each record.

**key-sequenced data set.** A VSAM data set whose records are loaded in key sequence and controlled by an index.

**keyword.** A part of a control statement that consists of a specific character string.

**KSDS.** Key-sequenced data set.

**LAN.** Local area network.

**LAN gateway.** A functional unit that connects a local area network with another network using different protocols.

**LDS.** Linear data set.

**LEN node.** Low-entry networking node. That is a node that provides a range of end-user services, attaches directly to other nodes using peer protocols, and derives network services implicitly from an adjacent APPN network node, that is, without the direct use of CP-CP sessions.

**linear data set.** A VSAM data set that contains data but no control information. A linear data set can be accessed as a byte-string in virtual storage. A linear data set has no records and a fixed control interval size of 4096 bytes.

**local.** Refers to one's own system.

**local area network.** A data network located on the user's premises in which serial transmission is used for direct data communication among workstations.

**local filestore.** A collection of local files. Each local filestore is registered in the LRD with a filestore nickname for local access, and a filestore AET for remote access.

**Local Resource Directory.** The file containing information on local users, local filestores, filestore subsets, and local files necessary for OSI/File Services to run initiator and responder functions.

**local-request handler.** A server subtask that can process a request submitted at the local system and can initiate a conversation.

**log file.** A file to which a NetView FTP component writes messages.

**logical unit.** In SNA, a port through which an end user accesses an SNA network. Each NetView FTP server is a logical unit.

**logical unit name.** A name used to represent the address of a logical unit.

**LRD.** Local Resource Directory.

**LU.** Logical unit.
**LU name.** Logical unit name.

**LU 0 conversation.** The type of conversation NetView FTP uses for file transfers between a node where NetView FTP V2.1 MVS, NetView FTP V1 VM, or NetView FTP V1 VSE is installed and a node where NetView FTP V1 MVS or FTP V2 is installed.

**LU-LU session.** In SNA, a session between two logical units (LUs) in an SNA network.

**manual transfer restart.** NetView FTP's method for allowing a user to restart a file transfer that was interrupted by submitting a restart request for that file transfer.

**master password.** A password, set by the NetView FTP system programmer that lets those who specify it query, modify, or delete any request in the request queue, regardless of whether or not it is password-protected, and regardless of who added it to the queue.

**message area.** The area of storage to which NetView FTP writes the messages it issues to an application program.

**mode.** The session limits and common characteristics of the session associated with advanced program-to-program (APPC) devices managed as a unit with a remote location.

**mode description.** A system object created for advanced program-to-program (APPC) devices that describes the session limits and the characteristics of the session, such as the maximum number of sessions allowed, maximum number of conversations allowed, and other controlling information for the session.

**MVS node.** A node with MVS as its operating system.

**MVS system.** A system with MVS as its operating system.

**NETBIOS.** Network Basic Input/Output System. An operating system interface for application programs used on IBM personal computers that are attached to the IBM Token-Ring Network.

**NetView FTP administrator.** Someone who knows the master password. A NetView FTP administrator can query, delete, modify, hold, or release any request regardless of whether it is password-protected, and regardless of who submitted it.

**NetView FTP AIX.** NetView File Transfer Program Server for AIX and NetView File Transfer Program Client for AIX.

**NetView FTP application program.** An application program that adds, queries, modifies, or deletes a request, or that retrieves information about NetView FTP.

**NetView FTP batch job.** A batch job that adds, queries, modifies, or deletes a request, or that retrieves information about NetView FTP.

**NetView FTP Client for DOS and Windows.** NetView File Transfer Program &clw.

**NetView FTP MVS.** NetView File Transfer Program for MVS.

**NetView FTP OS/2.** NetView File Transfer Program Server/2 and NetView File Transfer Program Client/2.

**NetView FTP/400.** NetView File Transfer Program for OS/400.

**NetView FTP partition.** A VSE partition that contains the main components of NetView FTP VSE.

**NetView FTP shared block.** An area of CSA storage that is used to pass data between the components of NetView FTP. Any data that does not fit in the FSB is put in the NetView FTP shared block extension.

**NetView FTP shared block extension.** An area of ECSA storage that is used to pass data between the components of NetView FTP. It contains any data that does not fit in the NetView FTP shared block.

**NetView FTP VM.** NetView File Transfer Program for VM.

**NetView FTP VSE.** NetView File Transfer Program for VSE.

**network.** An interconnected group of nodes.

**network drive.** With NetView FTP, it is a shared resource that can be accessed from each workstation in the LAN.

**network job entry facility.** A facility that uses the network job-interface (NJI) protocols to allow a computer system to communicate with other computer systems in a network.

**NFTP directory.** The directory that contains the NetView FTP/2 product files.

**NFTPWORK directory.** The directory that contains all NetView FTP/2 work files, for example, the NetView FTP message and log files.

**NJE.** Network job entry.

**NJI.** Network job-interface.
node. An endpoint in a link, or a junction common to
two or more links in a network. A deprecated term for
server group.

node ID. Deprecated term for server group.

node ID table. Deprecated term for server group table.

not-after time. The time after which NetView FTP is
not to process a request.

not-before time. The time before which NetView FTP
is not to process a request.

numeric literal. A numeric character or string of
numeric characters whose value is implicit in the char-
acters themselves; for example, 777 is the literal as well
as the value of the number 777.

octal. Pertaining to a selection, choice, or condition
that has eight possible different values or states.

OEM. Original equipment manufacturer.

Open Systems Interconnection. The seven-layer
communications architecture used for the definition of
protocol standards for networks.

operation mode. See attended mode and unattended
mode.

operational key. Deprecated term for cryptographic
key.

Original equipment manufacturer. A manufacturer of
equipment that may be marketed by another manufac-
turer.

originator ID. A string of characters that identifies the
job, started task, or user that added a request to the
request queue.

OSI. See Open Systems Interconnection.

other-domain resource. A recommendation for a
logical unit that is owned by another domain and is ref-
erenced by a symbolic name, which can be qualified by
a network identifier.

panel. A predefined image displayed on a terminal
screen.

panel flow. The way in which panels are chained
together so that a user can move from one to another.

panel layout. The way in which the text and the input
fields on a panel are arranged.

partitioned data set. A data set in direct access
storage that is divided into partitions, called members,
each of which can contain a program, part of a
program, or data.

PCF. Programmed Cryptographic Facility.

PDS. Partitioned data set.

path information unit. In SNA, a message unit con-
sisting of a transmission header (TH) alone, or a TH fol-
lowed by a basic information unit (BIU) or a BIU
segment.

phase. The smallest unit of executable code that can be
loaded into virtual storage.

ping. The use of the ping command to send an echo
request to a network host or gateway.

port. With NetView FTP, the communication end point
in TCP/IP. A port is identified by a port number.

port number. In TCP/IP, a 16-bit number used to
communicate between TCP/IP and a higher-level pro-
tocol or application.

post-conversation user exit. A user exit that passes
control to a routine at the system at which the file-
transfer request originated. This routine is to run just
after the servers terminate their conversation.

post-transfer user exit. A user exit that passes
control to a routine that is to run just after a server
closes a file that has been transferred.

pre-queuing user exit. A user exit that passes control
to a routine that is to run just after a request is sub-
mited to NetView FTP and just before NetView FTP
adds the request to the queue.

pre-transfer user exit. A user exit that passes control
to a routine that is to run just before a server opens a
file that is to be transferred.

preparation user exit. Deprecated term for pre-
transfer user exit.

process (a request). To obtain and try to carry out (a
request).

program temporary fix. A temporary solution to
bypass of a problem diagnosed by IBM as resulting
from a defect in a current unaltered release of a
program.

PTF. Program temporary fix.

PUBX. Physical Unit Control Block Extension.

QDA. Query data area.

QRA. Query response area.
QSAM. Queued sequential access method.

QSR. Query Status Record.

query (a request). To ask for information about (a request).

query data area. The area of ECSA storage into which NetView FTP places either user query records or administrator query records.

query response area. The area of storage into which NetView FTP places information it retrieves for an application program.

queue handler. A NetView FTP component that controls access to the request queue. In NetView FTP MVS, the queue handler also controls all other NetView FTP components and all communication with the operator.

queue handler command. A command that a component of NetView FTP issues to the queue handler when it wants the queue handler to do something.

queued sequential access method. An extended version of the basic sequential access method (BPAM). When this method is used, a queue is formed of (1) input data blocks that are awaiting processing or (2) output data blocks that have been processed and are awaiting transfer to auxiliary storage or to an output device.

raw device. In the AIX operating system, a device that treats data I/O as a continuous stream, without consideration for the data’s logical structure. For example, I/O for fixed disks and streaming tapes occurs in units of bytes that have no relationship to characters.

RCB. Request control block.

RCE. Request control element.

RDF. Request definition file.

reason code. A value issued by a program that gives additional information about a situation described by a return code.

receiving data set. A data set in which a copy of a file that has been sent using NetView FTP has been placed. Contrast with sending data set.

receiving file. A data set in which a copy of a file is placed that has been sent using NetView FTP. Contrast with sending file.

receiving system. The NetView FTP system that receives the file being transferred.

regular checkpointing. To take a checkpoint at specified regular intervals.

regular file. In the AIX operating system, a file that contains data. A regular file can be a text file or a binary file. Text files contain information readable by the user. This information is stored in ASCII. Binary files contain information readable by the computer.

relative record data set. In VSAM, a data set whose records are loaded into fixed-length slots and are represented by the relative-record numbers of the slots they occupy.

remote. Pertaining to a system other than one’s own.

remote filestore. A collection of remote files. OSI/File Services users can access a file residing in a remote system only if they specify the filestore AET of the remote filestore, and the OSI file name of the related file.

remote-request handler. A server subtask that can accept a conversation initiated by a local-request handler at a remote system.

report. Synonym for file-transfer report.

report recipient. A user to whom a server sends a file-transfer report.

request. Synonym for file-transfer request.

request class. A deprecated term for server class.

request control block. A control block that contains some or all of a file-transfer request. Data that does not fit in the RCB is put in the request control block extension.

request control block extension. A control block that contains data that does not fit in the request control block.

request control element. An element of the request queue directory.

request database. In NetView FTP/400, the database in which NetView FTP stores all requests.

request definition file. A file containing NetView FTP control statements, that is, all NetView FTP parameters with their appropriate values that are necessary for a file transfer.

request handler. A server subtask. See also remote-request handler.

request number. A number that the queue handler assigns to a request when it adds the request to the request queue and that is used to identify the request.
**request password.** A character string, assigned by a user to a request, that prevents users sharing that user’s originator ID from deleting or modifying that user’s waiting and active requests. If a user specifies a request password for a request, another user with the same originator ID must specify either the request password or the master password to be able to delete or modify the request while it is waiting, or to delete the request while it is active.

**request priority.** A number, assigned by a user to each request, that determines the order in which a server is to process it. When a server is ready to process a request and several requests are eligible, the server processes the request with the highest priority first.

**request queue.** In NetView FTP for MVS, VSE, or VM, the file in which NetView FTP stores requests that have been submitted for processing.

**request queue directory.** A directory of the contents of the request queue.

**request unit.** A message unit that contains control information, end-user data, or both.

**request-queue user exit.** Deprecated term for pre-queuing user exit.

**requesting system.** The system where the file-transfer request has been initiated.

**resident session partner.** An FTP V2 MVS or FTP V2 VSE server that runs continuously and can perform an indefinite number of file transfers, however, these file transfers must all be initiated by servers at remote systems.

**responding system.** The system responding to a file-transfer request.

**return code.** A value issued by a program that describes the outcome of an operation performed by that program.

**root.** In the AIX operating system, the user name for the system user with the most authority.

**RRDS.** Relative record data set.

**RTM.** Recovery Termination Manager.

**RU.** Request unit.

**RXB.** Request control block extension.

**SAF.** System authorization facility.

**SAM.** Sequential access method.

**SAS.** Spool Access Services.

**saved request.** A request that has been created with the NetView FTP panels and that has then been saved in an ISPF table data set.

**scheduling a request.** Determining which request is to be obtained, obtaining it, and passing it to a server.

**scroll amount.** The amount that the list on a panel is scrolled up or down when you enter the **UP** or **DOWN** command.

**SDA.** Server data area.

**SDMU.** Source description message unit.

**SDWA.** System Diagnostic Work Area.

**select members.** To choose those members of a PDS that are to be transferred. Contrast with exclude members.

**sending data set.** A data set, a copy of which is to be transferred using NetView FTP. Contrast with receiving data set.

**sending file.** A file of which a copy is to be transferred using NetView FTP. Contrast with receiving file.

**sending system.** The NetView FTP system stores the file that will be transferred to the receiving system.

**sequential access method.** See basic sequential access method.

**server.** (1) A NetView FTP component that establishes or accepts conversations and that transfers files. (2) With NetView FTP for Workstations, a NetView FTP Server program that serves as a gateway for one or more NetView FTP Client programs.

**server class.** A number or letter, assigned by a user to a request, that specifies which servers can process that request.

**server data area.** The area of ECSA into which NetView FTP places information about a server or servers.

**server group.** A group of servers (logical units).

**server group table.** A data set that specifies which servers make up each server group, and that contains information about each server.

**server modification area.** An area of storage that contains the modifications to a server’s session parameters.
**session.** In SNA, a logical connection between two network-addressable units.

**SFS.** Shared file system.

**SFS directory.** A group of files. SFS directories can be arranged to form a hierarchy in which one directory can contain one or more subdirectories as well as files.

**shared file pool.** See file pool.

**shared file system.** A part of CMS that lets users organize their files into groups known as directories, and selectively share those files and directories with other users.

**single mode.** A server running mode in which a server stops running after it has transferred a file (or attempted to transfer a file and failed).

**slot.** A space in the request queue directory that is able to hold a request control element (RCE).

**SMA.** Server modification area.

**SMF.** System management facilities.

**SMS.** Storage Management Subsystem.

**SNA.** Systems Network Architecture.

**SNA compaction.** The transformation of data by packing two characters in a byte that normally would only hold one character.

**SNA compression.** The replacement of a string of up to 64 repeated characters by an encoded control byte to reduce the length of the string.

**SNA network.** In SNA, the part of a user-application network that conforms to the formats and protocols of Systems Network Architecture. It enables reliable transfer of data among end users and provides protocols for controlling the resources of various network configurations. The SNA network consists of network addressable units, boundary function components, and the path control network.

**SNA node.** A node that supports SNA protocols.

**socket.** In the AIX operating system: (1) A unique host identifier created by the concatenation of a port identifier with a TCP/IP address. (2) A port identifier. (3) A 16-bit port number. (4) A port on a specific host; a communications end point that is accessible through a protocol family's addressing mechanism. A socket is identified by a socket address. See also socket address.

**socket address.** In the AIX operating system, a data structure that uniquely identifies a specific communication end point. A socket address consists of a port number and a network address. It also specifies the protocol family.

**SRMU.** Statistics report message unit.

**special file.** In the AIX operating system, a file that defines a FIFO (first-in, first-out) file or a physical device.

**SSCP.** System Services Control Point.

**statistics report message unit.** A message unit sent by the target (receiver of the file), at the end of the file transfer.

**status.** The state of a request in the request queue (MVS, VSE, or VM) or request database (OS/400). In NetView FTP for MVS, VSE, or VM, the possible statuses are waiting, active, and finished. In NetView FTP/400, the possible statuses are held, waiting, active, finished, and failed.

**status data area.** The area of ECSA into which NetView FTP places details of the status of a server or servers.

**Storage Management Subsystem.** An MVS subsystem that helps automate and centralize the management of DASD storage. SMS provides the storage administrator with control over data class, storage class, management class, storage group, and ACS routine definitions.

**submit (a request).** To give (a request) to NetView FTP so that NetView FTP can add it to the request queue.

**superuser.** In the AIX operating system, the user who has unrestricted authority to access and modify any part of the operating system, usually the user who manages the system.

**supervisor call instruction.** An instruction that interrupts a running program and passes control to the supervisor so that the supervisor can perform the service indicated by the instruction.

**SVA.** Shared Virtual Area.

**SVC.** Supervisor call instruction.

**symbolic constant.** A data item that has an unchanging, predefined value.

**system authorization facility.** At an MVS or VM location, a generic interface to security products that is provided by the operating system. In this way, an installation has the possibility to run a security program, such as IBM RACF.
System management facilities. An optional control program for MVS that provides the means for gathering and recording information that can be used to evaluate system usage.

System services control point. In SNA, a focal point within an SNA network for managing the configuration, coordinating network operator and problem determination requests, and providing directory support and other session services for end users of the network. Several SSSPs, cooperating as peers, can divide the network into domains of control, with each SSCP having a hierarchical control relationship to the physical units within its domain.

Systems Network Architecture. The description of the logical structure, formats, protocols, and operating sequences for transmitting information units through, and for controlling the configuration and operation of, networks. The layered structure of SNA allows the origin and ultimate destination of information to be independent of and unaffected by the SNA network services and facilities used to transfer that information.

table display panel. A panel that contains a scrollable list.

Target Information Message Unit. A message unit sent by the receiving node after it receives an SDMU. A TIMU updates the original request, to inform the sender where to start or restart sending, and, to convey statistical information.

TCP. Transmission Control Protocol.

TCP/IP. Transmission Control Protocol/Internet Protocol. A set of communication protocols that support peer-to-peer connectivity functions for both local and wide area networks.

TIMU. Target Information Message Unit.

token-ring network. A network that uses a ring topology, in which tokens are passed in a circuit from node to node. A node that is ready to send can capture the token and insert data for transmission.

Transfer Request Message Unit. A message unit used to send or receive a file, or to restart a previously interrupted file transfer.

Transmission Control Protocol. A communications protocol used in Internet and in any network that follows the U.S. Department of Defense standards for internetwork protocol. TCP provides a reliable host-to-host protocol between hosts in packet-switched communications networks and in interconnected systems of such networks. It assumes that the Internet protocol is the underlying protocol.

TRMU. Transfer Request Message Unit.

unattended mode. An operating mode of NetView FTP for Workstations that assumes that a user is not available at the workstation, for example, to load devices such as diskettes or tapes. Contrast with attended mode.

unused slot. A slot that does not contain a request control element (RCE).

user exit. A point in an IBM-supplied program at which a user-exit routine is given control.

user interface. Hardware, software, or both that allows a user to interact with and perform operations on a system or program.

user query command. A type of command that causes NetView FTP to retrieve information about the requests in the request queue; the information retrieved is intended for users of NetView FTP. Contrast with administrator query command.

user query record. A record containing the information NetView FTP retrieves when a user issues a user query command. Contrast with administrator query record.

user-exit routine. An exit routine written by a user.

user-written file handler. An exit routine, written by a user, that gains access to a file and passes its records to NetView FTP, or that retrieves the records from NetView FTP and writes them to a file.

virtual storage access method. An access method for indexed or sequential processing of fixed- and variable-length records on direct-access storage devices. The records in a VSAM data set can be organized (1) in logical sequence by means of a key field (key sequence), in the physical sequence in which they are written in the data set (entry sequence), or (2) by means of a relative-record number.


VM node. A node with VM as its operating system.

VM system. A system with VM as its operating system.

VSAM. Virtual storage access method.

VSE node. A node with VSE as its operating system.

VSE system. A system with VSE as its operating system.

waiting request. A request that is waiting to be processed.

WAN. Wide area network.

Wide area network. A network that provides communication services to a geographic area larger than that served by a local area network and that may use or provide public communication facilities.

wildcard character. Either a question mark (?) or an asterisk (*) used as a variable in a file name or file name extension when referring to a particular file or group of files.

WTO. Write-to-operator.

XID. Exchange identification.

XnMU. Exit(n) message unit.
Bibliography

The NetView FTP Library

This manual is part of a library of publications that describe NetView FTP and explain how to use it. The publications in this library are:

NetView File Transfer Program for VSE:
- NetView FTP Licensed Program Specifications, GH12-5485
- NetView FTP VSE Installation, Operation, and Administration, SH12-5674
- NetView FTP Customization, SH12-5482
- NetView FTP Messages and Codes, SH12-5483
- NetView FTP Parameter Reference, SH12-6052

NetView File Transfer Program for VM:
- NetView FTP General Information, GH12-5480
- NetView FTP Licensed Program Specifications, GH12-5485
- NetView FTP VM Installation, Operation, and Administration, SH12-5676
- NetView FTP Customization, SH12-5482
- NetView FTP Messages and Codes, SH12-5483

NetView File Transfer Program for OS/400:
- NetView FTP General Information, GH12-5480
- NetView FTP V1 for OS/400 Licensed Program Specifications, GH12-5777
- NetView FTP V3 for OS/400 Licensed Program Specifications, GH12-6176
- NetView FTP V1 for OS/400 Installation and User's Guide, SH12-5776

NetView File Transfer Program Version 2 for MVS:
- NetView FTP Licensed Program Specifications, GH12-5485
- NetView FTP V2 MVS Installation, Operation, and Administration, SH12-5657
- NetView FTP Customization, SH12-5482
- NetView FTP Messages and Codes, SH12-5483
- NetView FTP Parameter Reference, SH12-6052

NetView File Transfer Program Server for AIX, NetView File Transfer Program Client for AIX:
- NetView FTP Parameter Reference, SH12-6052

NetView File Transfer Program Server/2, NetView File Transfer Program Client/2
NetView File Transfer Program Client for DOS and Windows:
- NetView FTP Parameter Reference, SH12-6052

The unlicensed manuals with prefix SH are also available as softcopy on the following collection kits:
- IBM Networking Softcopy Collection Kit, SK2T-6012
- IBM Online Library Omnibus Edition: MVS Collection, SK2T-0710
- IBM Online Library Omnibus Edition: VM Collection, SK2T-2067
- IBM Online Library Omnibus Edition: VSE Collection, SK2T-0060
- IBM Online Library Omnibus Edition: AIX Collection, SK2T-2066

Related Publications

VTAM Installation and Resource Definition Version 3, SC23-0111
ACF/VTAM Version 3 Customization, SC23-0112
MVS/XA: System Macros and Facilities, Volume 2, GC28-1151
OS/VS2 SPL: Supervisor, GC28-1046
NetView FTP V2 MVS Program Directory
File Transfer Program Version 2: Diagnosis Reference, LY12-5350
TSO/E Version 2—REXX/MVS Reference, SC28-1883
SAA Common Programming Interface - C Reference, SC09-1308

21 NetView FTP/400 V1 only.
Index

A
abnormal termination 11
about this book xi
access
authority (OS/400) 132
authorization checking 6
checking 6
violation 6
accessing data
user-written file handler 3
address
disk 125, 131
disk, internal 125, 131
I/O area 6, 9
query response area 114
sending file ID 124, 130
space of, deleting the file handler from 3
amounts, block 128
APF authorized library 3
APL
description of 109
layout of 110
APL, application program parameter list 41
APLAPPWD 119
APLAPUID 119
APLAPXPT 46, 136
APLBYTC 134
APLBYTNCL 134
APLCHNGE 87
APLCLASS 40, 116
APLCLEAR 142
APLCOMPR 46, 138
APLCNTNT 138
APLCONVT 46
APLDELAY 117
APLDLRF 142
APLDYDE 46, 137
APLDYIE 46, 137
APLDYKL 46, 136
APLDYKO 46, 137
APLDYME 46, 137
APLDYMP 136
APLDYMPW 46, 137
APLDYRSA 46, 136
APLDYUCP 46, 137
APLDYUPL 136
APLFBALN 136
APLFBAP 114
APLFORUS 116
APLGXDLN 46, 56
APLGDPT 46, 56
APLID 110
APLJOE 135
APLLLUNM 119
APLLNGTH 111
APLXXXLN 46, 56
APLXXXDP 46, 56
APLMACT 115
APLMAPI 115
APLMSQLN 115
APLMTOT 115
APLNAATL 117
APLNBATL 117
APLNETID 40, 119
APLNODE 40, 118
APLORIG 116
APLPMCL 133
APLPMXR 134
APLPMXR 133
APLPPXIN 46, 56, 78, 119
APLPRTY 40, 116
APLPSMOD 133
APLPSWD 40, 117
APLPSWN 411, 123
APLPTIME 133
APLPTRCL 56, 133
APLPTRCP 56, 133
APLQACNT 114
APLQSTL 137
APLQTCNT 114
APLR4ACC 132
APLR4CIL 132
APLR4ICL 132
APLR4LCL 132
APLR4MCL 132
APLR4MCP 132
APLR4MRC 132
APLR4MXXI 132
APLR4MXM 132
APL4MNRC 132
APLR4NTY 132
APL4TXT 132
APLR4CC 46, 130
APLR4GCI 46, 126
APLRACPW 46, 126
APLRACU2 46, 126
APLRACUI 46, 126
APLRARCA 130
APLRAVBL 46, 128
APLRC 56, 111
APLRCCSI 46, 130
APLSDYDD 48, 122
APLSDYFD 48, 122
APLSDYLI 48, 123
APLSDYLIR 48, 123
APLSDYMM 48, 122
APLSDYPH 48, 123
APLSDYQU 48, 123
APLSDYRE 48, 122
APLSDYUC 48, 122
APLSDYUP 48
APLSDYUT 48, 123
APLSECV 48, 135
APLSEOF 124
APLSEOPS 48, 124
APLSEOR 48, 124
APLSFHMD 48, 124
APLSFLAB 48, 124
APLSFORG 48, 124
APLSFTYP 48, 122
APLSINGL 138
APLSJNO1 48
APLSJNO2 48
APLSJNO3 48
APLSJNO4 48
APLSJNO5 48
APLSJNO6 48
APLSJOK1 48
APLSJOK2 48
APLSJOK3 48
APLSJOK4 48
APLSJOK5 48
APLSJOK6 48
APLSJPD5 48
APLSSLFA 48, 124
APLSLFL 48, 124, 130
APLSNNID 48, 120
APLSNUID 48, 120
APLSOWID 48, 125
APLSPOID 49, 125
APLSRCE 139
APLSSMOD 138
APLSSUD1 49
APLSSUD2 49
APLSSUD3 49
APLSSUD4 49
APLSSUD5 49
APLSSUD6 49
APLSSUD7 49
APLSSUD8 49
APLSSUDn 125
APLSTOR 139
APLSVID 49, 125
APLSVEX3 49
APLSVEXT 49, 125
APLSVFML 49, 125
APLSVIN3 49
APLSVINT 49, 125
APLSVPW 49, 125
APLUEUI0 49, 120
APLUEUI1 49
APLUEUI2 49
APLUEUI3 49
APLUEUI4 49
APLUEUI5 49
APLUEUI6 49
APLUEUI7 49
APLUEUI8 49
APLUEUI9 49
APLUXLEN 49, 56, 78, 119
APLUXPTR 49, 56, 78, 119
APLVBC 112
APLVID 111
APLVTP 49, 136
APLXMODE 117
APLXOPT 49, 136
APPC user ID and password 119
application program parameter list (APL) 41
APX
  addressing 143
  Assembler 143
  C language 143
  identifier 141
  layout of 141
  PL/I 143
  REXX 143
  APXENT 49, 142
  APXID 141
  APXLEN 49, 142
  APXOPT 49, 142
  APXRNME 49, 142
  APXSNAME 49, 142
  APXSRCE 143
  APXSTOR 143
  APXTYP 49, 142
  APXXNME 49, 142
area
  address, I/O 6
  length, I/O 6
Assembling, user-exit routines 1
assembling the file handler 3
attribute
  archive file 130
  archive file attribute 130
  hidden file 130
  hidden file attribute 130
  read-only file 130
  read-only file attribute 130
  system file 130
  system file attribute 130
attribute, nonreusable 3
automatic restart parameter 118
automatic session recovery
close and reopen a file 11
implementing restart from checkpoint 19
OPEN function 19
automatic transfer restart 14, 20
average record (SMS parameter) 131

B
block
amounts 128
length 128
size 123, 128
blocks, number of directory 128
book, about this xi
buffers 18
byte checkpoint interval 29
BYTECR 18
bytes transferred
  compressed 134
  uncompressed 134

C
C language
  linking the user-exit routine 160
  user-exit routines 159
call
  GET function, file-handler 7
  POINT function, file-handler 10
  PUT function, file-handler 9
CALL argument list
  checkpoint record key area 30
  file-handler parameter area 30
  file-handler parameter list 30
  first checkpoint information area 31
  input/output area 30
  message area 30
  second checkpoint information area 31
  calling a user-written file handler 4
  calling the post-transfer user-exit routine 56
  calling the pre-transfer user-exit routine 50
  calling the user-written file handler 3
CEESTART 159
changing parameters
  post-conversation user-exit routine 78
  pre-transfer user-exit routine 46
character
  padding 130
  padding character 130
checking
  access 6
  access-authorization 6
checkpoint
  considerations 14, 15
  designing 14
  file 18
  in the file handler, designing 14
  information for the POINT function 10
  information, identical for sender and receiver 18
  information, PUT function 17
  information, update 20
  interval 29, 30
 PUT function 15
  record 14
  record key 29
  record key data 16
  restart from 5
checkpoint interval
  byte 29
  time 30
checkpoint record key indicator 29
checkpointing
  not using 13
class
  modifying request 39
  server 116
  trace 105, 106
cleanup-processing 11
close a file for automatic session recovery 11
CLOSE function
  file handler 11
  function code 11
  I/O area 12
  message to the partner node 11
  return codes 12
  statistical information 11
code
  for the CLOSE function 11
  for the OPEN function 6
  for the PUT function 9
  for the PUT function, return 10
  X'00' 12
  X'20' 12
  X'40' 12
Coded Character Set Identifier 124, 130
codes
  verb 111
coding guidelines for the pre-queuing user-exit routine 41
combining, pre-transfer and post-transfer user-exit routines 91
command, originator-ID of current 116
communicating
  between user-exit routines 81
compacting
  the file handler 3
  compress records 7
contents of the key provided by user-written file handlers 17
customizing 1
current command, originator-ID 116
deleting 3
deliberation record 124
disk address, internal 131
disk user ID 125, 131
disposition, output 128
deleting the file handler 3
deleting the file 18
devices, types supported 3
devices, interface with transfer program 5
development 4
devices, passivating 4

data

DD name 122
information 122
name 122, 126
type 127
date, expiration 129
date, of the data set 122
date, of the file 127
debugging 6
delay time 117
directory blocks, number of 128
disk address, internal 131

disk user ID 125, 131
disposition, output 128
DVGSPOOL 71
DVGMXXMT 71, 73
DVGSPOOL 71

example of how DVGEXJOB works in NetView FTP
VSE 68
examples of using DVGJOBS 61
exit routine input 119
exit routines, guidelines for 101
expiration date 129

feedback fields 111

fields

feedback 111
FUECURCD 6
user file-handler parameter list 26

file

access option 124, 130
ID 122, 126
information 122, 126
organization 124, 130
status option 130
text description (OS/400) 132
transfer communication system 1
type 122
types supported 3

file handler

accessing files 3
assembling 3
calling the user-written 3
CLOSE function 11
coding for serial reusability 3
compiling 3
contents of the key provided by user-written 17
deleing 3
designing checkpointing in 14
functions 4
GET function 7
guidelines 101
how information parameter values are presented 31
inability to protect 3
input 120
interface with transfer program 5
internal 3
invoking 3
link-editing 3
linking to an APF authorized library 3
load module name 123, 129
loading 3
loading a new copy 3
main skeleton 4
OPEN function 6
passing control to 3
PUT function 9
receiving 18
security parameter 6
sending 18
sequence of function calls 33

end-of-file option 124, 130
end-of-processing option 124, 130
enhancements 5
environment, parallel-transfer 3

ESTAE 11

D

data class (SMS parameter) 131
data request units, number of transferred 18
data set 22

D

data class (SMS parameter) 131
data request units, number of transferred 18
data set 22

D

data class (SMS parameter) 131
data request units, number of transferred 18
data set 22

D

data class (SMS parameter) 131
data request units, number of transferred 18
data set 22

D

data class (SMS parameter) 131
data request units, number of transferred 18
data set 22

D

data class (SMS parameter) 131
data request units, number of transferred 18
data set 22

D

data class (SMS parameter) 131
data request units, number of transferred 18
data set 22
file handler (continued)
  user parameters 5
  user-written 1
  using any kind of 3
file handler functions
  guidelines 5
  implementation 5
file handling
  mode 124, 130
file ID 122, 126
file name
  DD name 127
file processing
  option 130
file processing option 130
file status option 130
file transfer
  normal 33
  with automatic session recovery 35
file-handler call
  GET function 7
  POINT function 10
  PUT function 9
file-handler function
  CLOSE 4
  GET 4
  OPEN 4
  POINT 4
  PUT 4
file-handler interfaces, user-written 23
filemode, internal 125, 131
FIOWAD
  description 27
  FIOWAD 6, 9
  FUEIOLAN 6
  FUERECLN 9
  GET function 8
  OPEN function 6
  PUT function 9
FIOWLN 27
FMODE 26
format, record 122, 127
FTPSECUR 43, 99
FUECPIBC 18, 29
FUECPITS 18, 30
FUECRPI 27
FUECRPO 27
FUECRREQ 13, 29
FUECRREQ, PUT function 13
FUECURCD
  CLOSE function 11
  description 28
  GET function 7
  OPEN function 6
  POINT function 10
  FUEINPTR 26
  FUEIOLAN 6, 8, 28
  FUEKEYP 27
  FUEKNFTP 29
  FUEMSGP 28
  FUEPHN 27
  FUEPOINT 27
  FUEPRECD 11, 28
  FUEPRERC 11, 28
  FUERECLN 8, 9, 29
  FUERECCOV 13, 29
  FUERETCD
    CLOSE function 12
    description 28
    FUERETCD 12
    GET function 8
    OPEN function 7
    POINT function 11
    PUT function 10
  FUERSTPT 29
  FUERUCNT 26
  FUERUNCR 18, 26
  FUETRCL 28
  FUETRCP 28
  FUEXPLRO 26
  FUEXPLRW 28
  fullwords 30
  function
    checkpoint for PUT 15
    CLOSE 11
    CLOSE function 11
    file-handler call for CLOSE 11
    file-handler call for POINT 10
    file-handler call for PUT 9
    FIOWAD 8
    for the POINT function 11
    FUECURCD 11
    FUEIOLAN 8
    FUEPRECD 11
    FUEPRERC 11
    function code for CLOSE 11
    function code for POINT 10
    function code for PUT 9
    GET 7, 103
    GET function 7
    I/O area for CLOSE 12
    I/O area for GET 8
    I/O area for POINT 11
    I/O area for PUT 9
    of the file handler 4
    of the post-transfer user-exit routine 56
    of the pre-queuing user-exit routine 38
    of the pre-transfer user-exit routine 45
    OPEN function 6
    OPEN, function code 6
    POINT 5, 10, 17
function (continued)
  POINT function 10
  purpose of PUT 9
  PUT function 9
  return codes for CLOSE 12
  return codes for GET 8
  return codes for PUT 10

G
GET function 7, 8, 103
glossary 163
group ID, RACF 121
guidelines for exit routines 101

I
I/O area
  address 9
  for the CLOSE function 12
  for the GET function 8
  for the POINT function 11
  for the PUT function 9
  length 6
  reinitializing 8
  retrieving the length 8
ID
  sending file 124, 130
identifier
  APLID 110
  APX 141
indicator, job submission 135
indicator, quote 123, 129
indicator, security verified 135
initial number of records (OS/400) 132
input
  exit routine 119
  file handler 120
  parameters 61, 65
installing
  DVGEXJOB 65
  DVGMXXMT 74
  DVGSPool 72
interface
  between the transfer program and file handler 5
  coding guidelines, user-exit routine 41
  file handler and transfer program 23
  user-written file-handler 23
interface routine 1
internal
  disk address 125, 131
  file handler 3
  filemode 125, 131
invoking
  a trace 107
  DVGEXEC 70

invoking (continued)
  DVGEXJOB 65
  DVGMXXMT 76
  DVGSPool 73

J
job submission indicator 135
job submission routine
description 64

K
key by user-written file handlers 17
key data, what to specify 16
key of record, for sending file handler 18
key, creating 16
KSDS, VSAM 127

L
label
type 124, 130
layout of the user file-handler parameter list 23
length
  block 128
  I/O area 6
  logical record 123, 128
  of area that the transfer program is to trace 133
  of message area 115
  of query response area 114
  of the I/O area, retrieving 8
  record 9
  record, GET function 8
  sending file ID 124, 130
library
  APF authorized 3
  like (SMS parameter) 131
link
  parameters 125, 131
  password 125, 131
list of volume serial names 123, 129
load module name
general 3
  of a user-written file handler 123, 129
loading a new copy of the file handler 3
local LU-name 119
logical record length 128
logical record size 129
LU-name
  local 119
  modifying 39
  remote 118
M
main skeleton of file handler  4
management class (SMS parameter)  131
manual transfer restart  14, 20, 22
maximum increments (OS/400)  132
maximum members per file (OS/400)  132
member
  name, PDS  122
  option (OS/400)  132
message
  area  101
  for a nonzero return code  103
routing  103
message area
  from user-exit routines  101
  length  115
  number of the messages  115
  pointer  115
  response area  114
migrating
  a user-written file handler to NetView FTP V2.2
    MVS  36
mode
  file handling  124, 130
modification of submitted jobs  62
MODIFY function  138
modifying
  a request password  38
  a server group name  39
  an LU name  39
  server class  39
  wait time  40
module name of a user-written file handler, load  123, 129
module name, load  3
MVS/XA
  file handler  2
  post-conversation user-exit routine  2
  post-transfer user-exit routine  2
  pre-queuing user-exit routine  2
  pre-transfer user-exit routine  53
new password  117, 123
node ID for program at sending system  120
node, transfer  117
nonreusable attribute  3
nonzero return code, messages  103
normal file transfer  33
not-after time and date  117
not-before time and date  117
number
  of bytes transferred, compressed  134
  of bytes transferred, uncompressed  134
  of directory blocks  128
  of messages in message area  115
  of records per increment (OS/400)  132
  of records transferred  134
  of status information items retrieved  114
  of status items retrieved  114
  of transferred request units  133
  request  115

O
OPEN function
  file handler  6
    FUEREfov  13
  function code  6
option
  end-of-file  124, 130
  end-of-processing  124, 130
  file access  124, 130
  file processing  130
  file status  130
organization
  file  124, 130
originator-ID of current command  116
OS/400
  file name  125, 132
  file type  125, 132
  library name  125, 132
  member name  125, 132
output
  disposition  128
  tape density  129

P
parallel-transfer environment  3
parameter
  automatic restart  118
  checking  45
  link  125, 131
  of a file-transfer request  3
  receiving file  126
  restart from checkpoint  118
  restart point  118
  security  121, 126
name
  data set  122, 126
  load module  3
  PDS member  122, 126
  VSAM catalog  122, 127
name of a user-written file handler, load module  123, 129
needs of a file transfer communication system  1
NetView FTP file transfer, special features  3
network ID
  remote  119
parameter (continued)
  server initialization 18
partner node, CLOSE function message 11
password
  APLPSWD 117
  link 125, 131
  modifying a request 38
security parameter 121
  VSAM 122
  VSE 123
PDS member name 122, 126
period, retention 129
PL/I, user-exit routines 1
POINT function
  automatic session recovery 7
  call, PUT function requests 9
  file-handler call 10
  I/O area 11
pointer to message area 115
pool ID 125, 131
post-conversation user-exit routine
  changing parameters 78
  description 77
  general 1
  MVS/XA 2
post-processing 56
post-transfer user exit
  combining 91
post-transfer user-exit routine
  APL 57
  calling 56
  coding guidelines 56
  description 55
  functions 56
  general 1
  messages 59
  MVS/XA 2
  register usage 58
  return codes 59
  returning to transfer program 58
  saving the registers 58
  serialization 57
  traces 59
user-specific information 58
PPEXIT initialization parameter 91
pre-queuing user-exit routine
  APL 41
  class modification 39
  coding guidelines 41
  functions 38
  general 1
  LU name modification 39
  MVS/XA or MVS/ESA 2
  password modification 38
  register usage 41
  return codes 42
pre-queuing user-exit routine (continued)
  returning to transfer program 42
  server group name modification 39
  wait time modification 40
pre-transfer user exit 91
pre-transfer user-exit routine
  APL 51
  calling 50
  changing parameters 46
  checking parameters 45
  coding guidelines 50
  description 43
  function codes 50
  general 1
  guidelines 102
  messages 53
  MVS/XA or MVS/ESA 2
  register usage 51, 52
  return codes 52
  returning to transfer program 52
  serialization 50
  traces 53
  user-specific information 51
preparation activities 45
primary disk space quantity 128
priority, request 116
process request for another user 116
program at sending system, node ID 120
purpose
  of customizing 1
  of the GET function 7
  of the POINT function 10
  of the PUT function 9
PUT function
  checkpoint 15
  file handler 9
  FUECRREQ 13
  function code 9
  purpose 9
  requests for a POINT function call 9
  update the checkpoint information 20
Q
QRA 114
QUERY ADM function 137
query response area
  address 114
  length 114
queue handler commands 37
quote indicator 123, 129
R
RACF 6
RACF, group ID 121
RACHECK 6
receiving
  file, parameters 126
  system 4
  system, running on 5
record
  checkpoint 14
  compact and compress records 7
  count 19
  definition (NetView FTP) 3
  definition (used access method) 3
  delimiter 124
  encrypt records 7
  format 122, 127
  frequency of writing checkpoint records 18
  interchange of checkpoint records 3
  key data, checkpoint 16
  key of, for sending file handler 18
  key, checkpoint record 14
  length 9
  length, GET function 8
  length, logical 123, 128
  logical record length 123
  longest expected during a file transfer 9
  number for receiving file handler 18
  receiving file handler, record number 18
  sending file handler, key of record 18
records
  compact and compress 7
  encrypt 7
recovery 7, 14, 19
  ABEND exit routine 5
  ABNEXIT exit routine 5
  abnormal-termination recovery routine 5
  automatic session recovery 29, 35
  close and reopen a file for automatic session recovery 11
  ESTAE 11
  example of automatic session 20
reentrant 57
reference DD statement (SMS parameters) 131
remote LU-name 118
remote network ID 119
remote workstation address 119
remote workstation nickname 119
reopen a file for automatic session recovery 11
request
  class, see server class 39
  for another user, process 116
  number 115
  password, modifying 38
  priority 116
resource access control facility 6
response and message area 114

restart
  automatic transfer 14
  manual transfer 14
restart from checkpoint
  conditions 20
  during an automatic session recovery 19
  general 5
  parameter 118
restart point
  beginning 29
  checkpoint 29
  parameter 118
retention period 129
retrieve the length of the I/O area 8
return code
  for the CLOSE function 12
  for the GET function 8
  for the PUT function 10
returning to transfer program 32, 42, 52
REXX
  application program 145
  calling NetView FTP V2 MVS 152
  control-block fields 145
  DVGCXFOF 147
  DVGCXRGRT 147
  DVGCXRPRT 147
  EXEC 69
  external functions 147
  invoking a user-exit routine 155
  linking the user-exit routine 156
  message area 156

S
sample post-transfer user-exit routine for NetView FTP
  VM to run a REXX EXEC 69
saving the registers 32, 42, 52, 58
secondary disk space quantity 128
security model generic attribute (SMS parameter) 132
security model profile name (SMS parameter) 131
security parameter, file handler 6
security parameters 121, 126
security verified indicator 135
security/statistics collection user exit 99
select argument 4
select-group 4
sending
  data set, repositioning 19
  system 4
  system, running on 5
sequence of function calls for a user-written file handler 33
server
  class 116
  class, modifying 39
  entry point of the transfer program 3
server (continued)
- group name 118
- group name ID 39
- initialization parameter 18
- invoking a user-written file handler 3
- parallel transfers on the same server 3
- session interrupt 19
- SFS
  - pool ID 125, 131
  - subdirectory level 125, 131
  - top directory 125, 131
- size
  - block 123, 128
- skeleton of file handler, main 4
- SMS parameters 131
- space unit 128
- statistical information 11
- statistics collection 56
- status items retrieved, number of 114
- storage class (SMS parameter) 132
- subdirectory level 125, 131
- SUEPL control block 99
- syntax of the exit routine input parameter 73, 76
- system
  - receiving 4
  - receiving, running on 5
  - sending 4

T
- tape density, output 129
- time
  - and date, not-after 117
  - and date, not-before 117
  - calling a user-exit routine 133
  - delay 117
  - wait 40
- time checkpoint interval 30
- top directory 125, 131
- trace
  - area 133
  - class 105
  - invoking 107
  - mask character 107
  - trace facility 105
- transfer
  - and submit a job 64
  - mode 117
  - normal file 33
  - on the same server, parallel 3
  - restart, automatic 14
  - restart, manual 14
  - with automatic session recovery 35
- transferred
  - records 134
  - request units 133

TRC initialization parameter 107
- type
  - data set 127
  - device 123, 129
  - file 122
  - file, OS/400 125, 132
  - label 124, 130

U
- unit, space 128
- update the checkpoint information, PUT function 20
- user file-handler parameter list
  - description 23
  - fields 26
  - general 5
  - layout 23
- user ID
  - disk 125, 131
  - first security parameter 121
- for program at sending node 120
- user-exit routines
  - Assembler 1
  - calling the post-transfer 56
  - communication 81
  - for NetView FTP VM to run a REXX EXEC 69
  - PL/I 1
  - post-conversation 1, 77
  - post-transfer 1, 55
  - pre-queuing 1, 37
  - pre-transfer 1, 43
  - sequence of calls 92
  - using the pre-transfer and post-transfer 91
- user-specific information
  - pre-transfer user-exit routine 51
- user-written file handler
  - accessing data 3
  - contents of the checkpoint record key 17
  - general 1
  - guidelines 101
  - interfaces 23
  - invoking 3
  - load module name 123, 129
  - migrating to NetView FTP V2.2 MVS 36
  - register usage 32
  - saving the registers 32
  - sequence of function calls 33
- user, process request for another 116

V
- verb code 111, 112
- violation, access 6
- volume count 129
- volume serial names, list of 123, 129
VSAM
  catalog name  122, 127
  KSDS  127
  password  122

W
  wait time
    modifying  40
  workstation
    address  119
    nickname  119
  writing a checkpoint record to the checkpoint file  18

X
  X'00', code  12
  X'20', code  12
  X'40', code  12
Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you mail this form to us, be sure to print your name and address below if you would like a reply.

You can also send us your comments using:

- A FAX machine. The number is: +49–7031–166609.
- Internet. The address is: gadlid@sdfvm1.vnet.ibm.com.
- IBMLink. The address is: SDFVM1(GADLID).
- IBM Mail Exchange. The address is: DEIBM3P3 at IBMMail.

Please include the title and publication number (as shown above) in your reply.

Name

Address

Company or Organization

Phone No.
Your comments, please ...
SH12-5482-06

IBM Deutschland Entwicklung GmbH
Information Development, Dept. 0446
Postfach 1380
71003 Boeblingen
Germany
DSMBEG323I STARTING PASS 2 OF 2.
+++EDF248W Page check: document requires more passes or extended cross-reference to resolve correctly. (Page 36 File: DVGC0B03)
DSMMOM397I '.EDFPGCK' WAS IMBEDDED AT LINE 320 OF '.EDFHEAD1'
DSMMOM397I '.EDFHEAD1' WAS IMBEDDED AT LINE 34 OF 'DVGC0B03'
DSMMOM397I 'DVGC0B03' WAS IMBEDDED AT LINE 67 OF 'DVGC0MST'