BCOM for NSK Guardian Version 5.1

User's Guide

Table of contents

Presenting BCOM

The current ETI-NET BCOM product allows bi-directional file transfers, backups/restores, report distributions and job submissions to take place between a variety of platforms, which are: MVS (IBM mainframe), NSK Guardian, and Windows NT.

Data transfers take place using either peer-to-peer protocol over SNA lines or devices, TCP/IP connections, or, if the FTP option is used, FTP protocol over TCP/IP.

This section introduces the basic concepts and facilities of the BCOM product, the features and the various user interfaces.

This section covers the following topics:

- What is BCOM?
- BCOM Concepts
- BCOM Standard Features
- BCOM NSK Guardian Specific Features

What is BCOM?

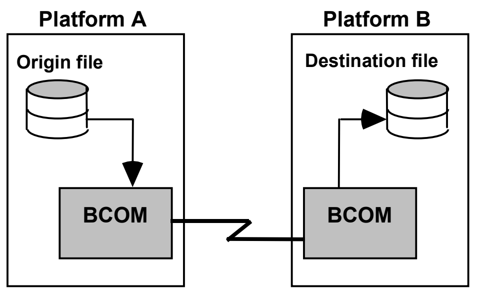
ETI-NET BCOM is a data transfer software used between heterogeneous platforms. It provides three standard functions: File Transfer, Job Submission and Print Distribution.

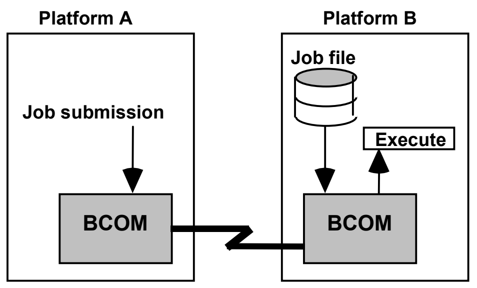
A fourth function is offered in option: Backup/Restore, which allows the use of the sophisticated data repositories of IBM/MVS as backup sites, from any BCOM platform.

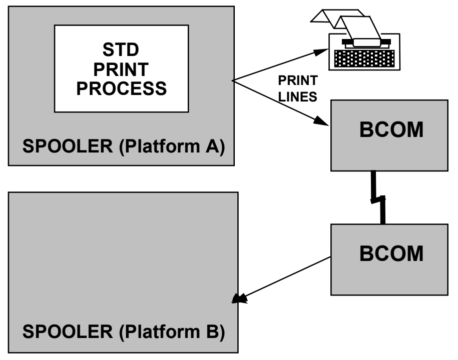
BCOM supports the following platforms: IBM/MVS, Tandem NSK Guardian, and Windows NT.

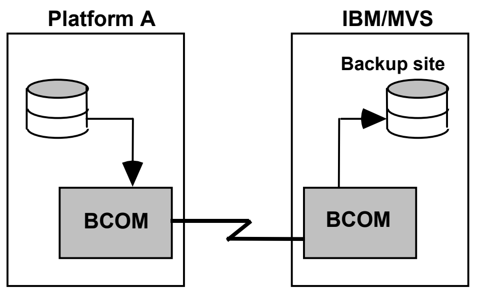
Transfers take place either across SNA lines or devices (including the channel attached SNAXLINK for NSK Guardian to MVS) or, across TCP/IP lines (using BCOM EndPoint facility), or as an FTP option across TCP/IP lines.

In addition, many features help in the defining, activating and controlling of transfers. The basic concepts and all features of BCOM are described in the rest of this chapter.

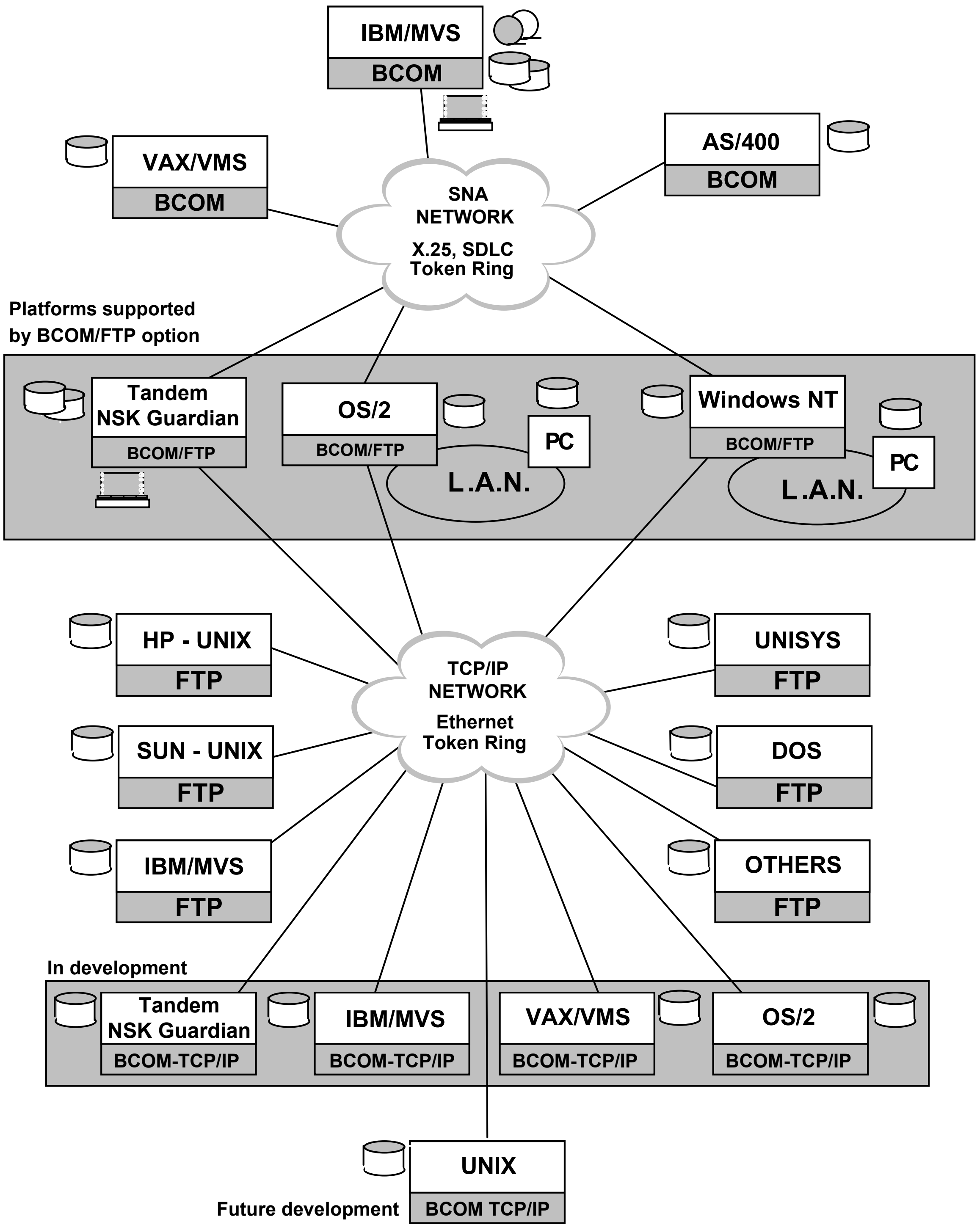








The BCOM NETWORK



BCOM Concepts

The Request

A request is used to define the attributes of a BCOM transfer task.

Each request is defined to BCOM through a series of commands. Once defined, the request may be added to the Request file, and once in the Request file, it request can be queued for execution.

No request definition is required in order to distribute print reports from spooler to spooler locations, using BCOM: report distribution takes place automatically and transparently.

The Queue

All requests must be queued for execution. Existing requests are queued for execution through the QUEUE command. Queued requests are placed in the Queue file, where BCOM will look in order to process the queued requests. If desired, requests may be queued based on a specific queuing priority.

The BCOM Interface (BFSINT)

BFSINT is the acronym used to describe the BCOM user interface. This interface can be used in both batch and interactive fashion on all supported platforms. The BFSINT interface is the vehicle through which all requests are defined and queued for the BCOM nucleus. In addition, all operation control commands are entered through BFSINT.

Local and Remote Locations

These terms have meaning only from the perspective of the initiator of a request. The request initiator refers to BCOM nodes in terms of the location from which the request is being defined and queued.

All references to LOCAL and REMOTE locations in commands are interpreted by BCOM as pertaining to valid location names defined in the local configuration file.

The LOCAL LOCATION is the BCOM node on which the BFSINT program executes and wherein a transfer request gets to be recorded into the Request file. When that node initiates a transfer with another node, the latter is called a REMOTE LOCATION.

The Requester Node

The requester node is the local or remote node that initiates the transfer request. The requester node contains the Request/Queue files on which the request is stored, and it is responsible for all transfer initiation tasks. BCOM implements this function through a Requestor task or process that is used to initiate transfers from the local location.

The Server Node

The server node is the node that accepts and processes the initiation messages sent by a requester node. The server node can be either a local or remote node. The requester node will give the server node the target file to read from or write to. The server node, when acting as a server, need not access its Request/Queue files. BCOM implements this function through a Server task or process, which processes transfer requests received from remote locations.

Note: *Remember that, globally speaking, all BCOM nodes, both local and remote, can act as both a requester node and a server node. The particular configuration decisions will determine the extent of these possibilities.*

The Configuration File

Each BCOM node must run with a configuration file. The configuration file describes the parameters that BCOM will run with on that node.

These parameters fall into two major classes: the parameters that define the environment of the LOCAL location and those that define the interface to each of the connectable REMOTE locations.

The following paragraphs outline some of the fundamental parameters of the configuration file that serve such purposes.

The Client (Requester) and Server processes

When a transfer request is activated, a client process - also called requester - communicates with a remote server process through either SNA or TCP/IP lines.

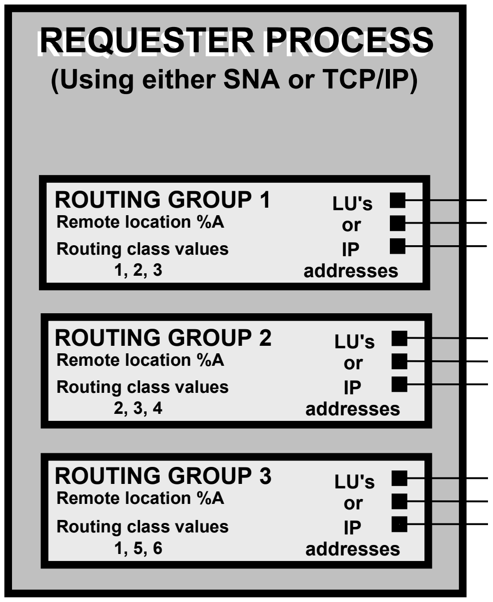
In order to allow parallel transfers, each client process can use up to 16 lines, all of which must use the same protocol (i.e. SNA or TCP/IP). This means that a unique client process could be linked to as much as 16 different server processes (provided each connection uses only one line), each of these servers being possibly on a different platform. However, lines can be grouped for connecting with a same remote server process. If used as a standard connection, such grouping allows for multiple concurrent transfers between a client and a server. Or it could be used to support the BCOM MULTIPLEX option: in multiplex mode, data is divided in parts that are sent simultaneously over all lines defined for a process, which means an important reduction of the transfer time.

Each server process can use up to 64 lines, all of which must use the same protocol (i.e. SNA or TCP/IP). The same rules as for client processes apply to servers, i.e. lines can be used for connections with different client processes that are possibly on different platforms, as well as they can be grouped for connecting with a same remote client process.

N.B.: *A maximum of 64 lines can be defined for a unique copy of BCOM.*

The Routing Classes

The physical paths (e.g. LU-LU connection or TCP/IP line) are defined inside process definitions, under the form of routing groups, each group referring to a line or a group of lines (all lines of a group using the same protocol, i.e. either SNA or TCP/IP). Each routing group is defined with a remote platform's name and a list of routing class values.



When the user defines a transfer request, he must set a routing class parameter, with an operand that must match one of the routing class values defined inside routing groups. Among the groups of lines that have this same value within their routing classes list, the first in order of definition and being available, is used.

By defining different routing class values inside separate routing groups with the same remote location name, the user can easily restrain the use of certain types of line (e.g. high-speed lines) to specific transfer requests. On the other hand, defining identical routing class values inside separate routing groups with the same remote location offers multiple paths for concurrent transfers. Note also that multiple lines in a routing group can be used either to allow concurrent transfers or to implement the BCOM MULTIPLEX option.

For example, if three routing groups are defined with a same remote location name, the first routing group being defined with routing values of 1, 2 and 3, the second group with routing values of 2, 3 and 4 and the third with routing values of 1, 5 and 6, then for all requests with this remote location name, those with a routing class value of 1 will be allowed to use the first or third routing groups (as these groups have 1 among their routing class values - the first available of these two groups after the queuing of such a request, is used), requests defined with routing class values of 2 or 3 will be allowed to use the first or second routing groups (because 2 and 3 are defined as routing class values in these two routing groups - the first available of these two groups after the queuing of such a re request, is used); requests with a routing class value of 4 will be allowed to use only the second routing group (because this group is the only one that has 4 among its routing class values); and finally, requests with a routing class value of 5 or 6 will be allowed to use only the third routing group (as only this group has 5 and 6 among its routing class values).

Notes: ***a)*** *BCOM will try to match a request's routing class value with the first defined and available group of lines (routing group), among groups that could be defined inside different requester processes, even if these processes refer to lines using different protocols (i.e. SNA or TCP/IP). However, all routing groups inside a same requester process must refer to lines using the same protocol.*

b) *Identical values can be used inside routing groups defined with different remote platforms, as BCOM limits its matching attempts to the routing groups defined with the same remote location as set in the transfer request definition.*

Automatic Follow-On Request Chaining

Requests can be initiated automatically based on the conditions of a previously defined and executed request. The conditions allow for the verification of a normal and/or abnormal end. For example, a second request can be set up so that it will be initiated only if the first one has ended normally. This is because, through a job submission request, a NOTIFY of the completion status can be sent to a process.

Automatic Conversion (ASCII/EBCDIC) of Selected Data

Using simple definitions, the user can specify which fields within each record are to be left un-translated. This feature is very useful for binary numbers embedded within records.

Conversion of Selective Signed Numeric Fields

Signed decimal fields are properly translated across the various platforms.

Data Compression and Decompression

Data compression and decompression can be set in BCOM to improve the effective data transfer rate and minimize telecommunications costs. When a file contains repetitive data (i.e. records padded with spaces or zeroes), use of this facility can considerably reduce the transfer time.

Flexible Interface

Commands to BCOM can be submitted interactively, in batch mode or through a programmatic interface.

Encryption

BCOM offers the possibility to encrypt automatically all data to be transferred. Decryption is made automatically on the destination platform.

BCOM Standard Features

Automatic Follow-on Scheduling

Requests can be initiated based on the conditions of a previously defined and executed request. The conditions allow for the verification of a normal and/or abnormal end. For example, a second request can be set up so that it will be initiated only if the first one has ended normally.

Command-line User Interface

Commands to BCOM can be submitted interactively, in batch mode or through a programmatic interface.

Conversion of Selective (ASCII/EBCDIC) Fields

Code conversion is accomplished automatically.

Using simple definitions, the user can specify which fields within each record are to be left untranslated. This feature is very useful for binary numbers embedded within records.

Conversion of Signed Numeric Fields

Signed decimal fields are properly translated across the various platforms.

Data Compression and Decompression

Data compression and decompression can be set in BCOM to improve the effective data transfer rate and minimize telecommunications costs. When a file contains repetitive data (i.e. records padded with spaces or zeroes), use of this facility can considerably reduce the transfer time.

Encryption

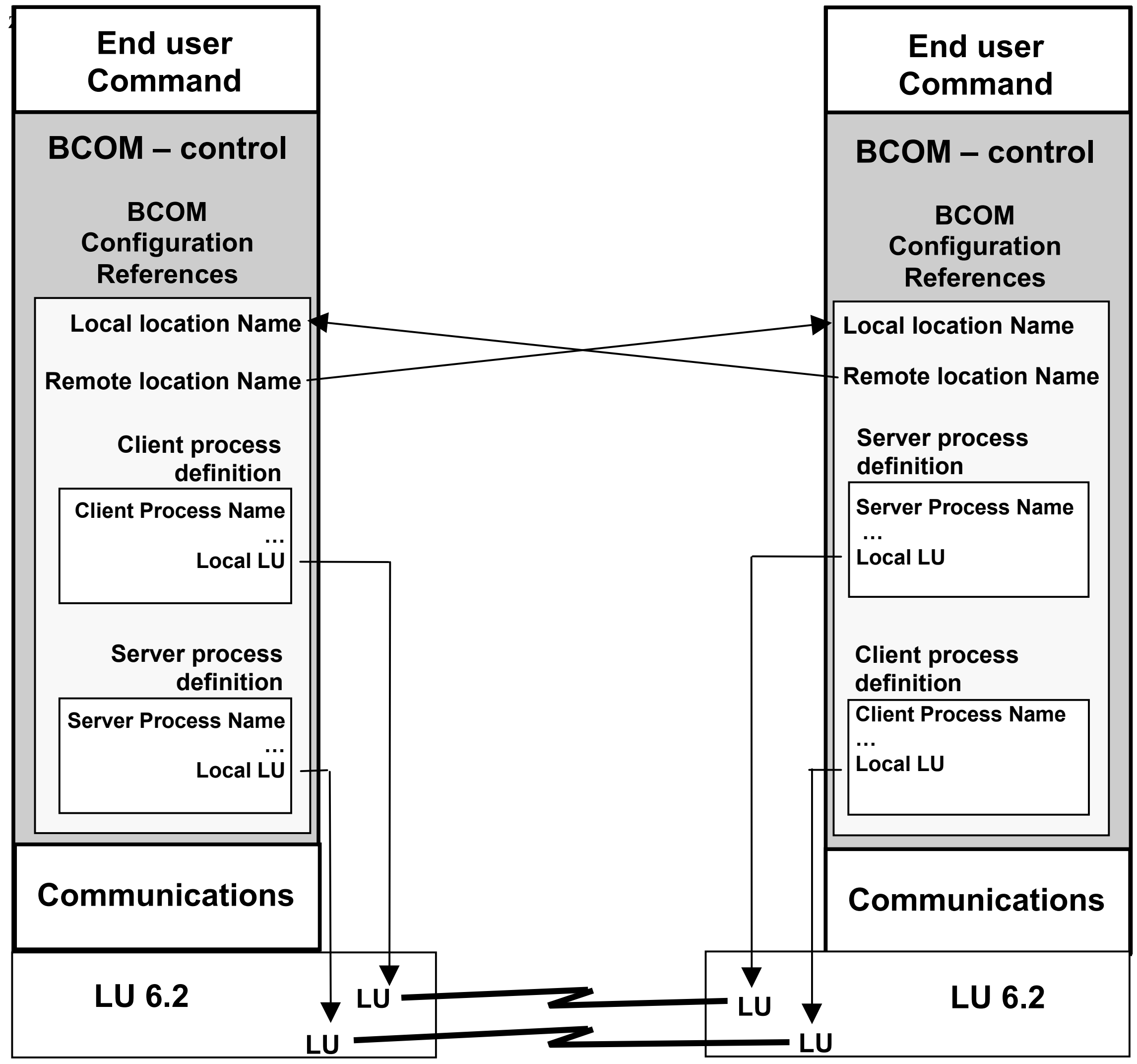
BCOM can be installed - or set - with an automatic encryption facility. Restoring of the encrypted data is made automatically and transparently on the destination platform whenever the data has been encrypted.

Job Report Distribution across Environments

Reports can be initiated for transfer from any platform. Both Spooler to Spooler and File to Spooler support are available.

Layered Architecture

As shown in the figure below, the layered architecture of BCOM provides a consistent user interface across all supported platforms:



Job Submission across Environments

BCOM allows a local node to submit jobs to any of its target nodes. Support is available for NSK Guardian's TACL, MVS's JCL, and Windows NT's .exe and .bat.

NSK Guardian supports WAITED job submissions. TACL type jobs can be submitted locally or from a remote with WAITED option, which would cause the request to terminate only after the completion of the job.

Logging Facility

All events regarding transfers or relating to the state of the environment are logged. Statistics are provided for all file transfers.

Queue Management Capabilities

All requests in the BCOM queue can be managed as desired. Routing classes can be specified, priority scheduling can be initiated, and queued requests can be held, removed or released. Remote queuing is accomplished by the RQUEUE command, and Follow-on Scheduling can be controlled conditionally to the completion status (normal, abnormal) of a process.

Auto Restart of Failed Requests

Auto Restart is a request definition feature that causes BCOM to keep a failed request in the queue file until the communication resources are once again available; then, BCOM automatically restarts the request from the beginning—or from the last valid checkpoint if that feature was also specified.

It can be used to recover from failures of communication, hardware and software components (including BCOM) whereas other facilities (such as the ABNORMAL-END) are designed to handle request failures — e.g. file not found, allocation errors, etc.

Report Distribution Across Environments

Reports can be initiated for transfer from any platform. Both Spooler to Spooler and File to Spooler support are available.

Remote Queueing Facility

BCOM provides a method for initiating already defined requests on a remote location. This method, called Remote Queuing Facility (RQF), does not rely on indirect mechanisms such as Job Submission or remote logons. Instead, it introduces a new type of request definition (type = RQ), which allows for the propagation of queue commands to remote nodes.

The RQF can be used to broadcast control throughout a network by enabling immediate node queuing. This can be used advantageously to automate the distribution of software or for the distribution of corporate information to all of your platforms.

Statistic Gethering

The Request statistic file contains all information needed to evaluate the user's amount of data transmitted, usage of communication bandwidth, increase in backup volume etc…

The data in the statistics file can be accessed by the Tandem utility ENFORM or it can be loaded into a SQL table.

Security

BCOM security is integrated to the sub-system of each platform except OS/2, for which ETI-NET provides an access profiles definition utility. A special security file is used on each BCOM platform Windows NT, to provide signon, access and command-level security for correlated user ids. A Security Table Manager on every platform except Windows NT provides management of those security files. On Windows NT, security definitions are set at the NT Services level.

User Defined Checkpoint/Restart

Large file transfers can be checkpointed at specific record intervals, thus enabling transfers to be restarted from the last valid checkpoint in case of critical errors. This can provide considerable savings in the time required for retransmission, as the data already transferred up to the last valid checkpoint will not be retransmitted.

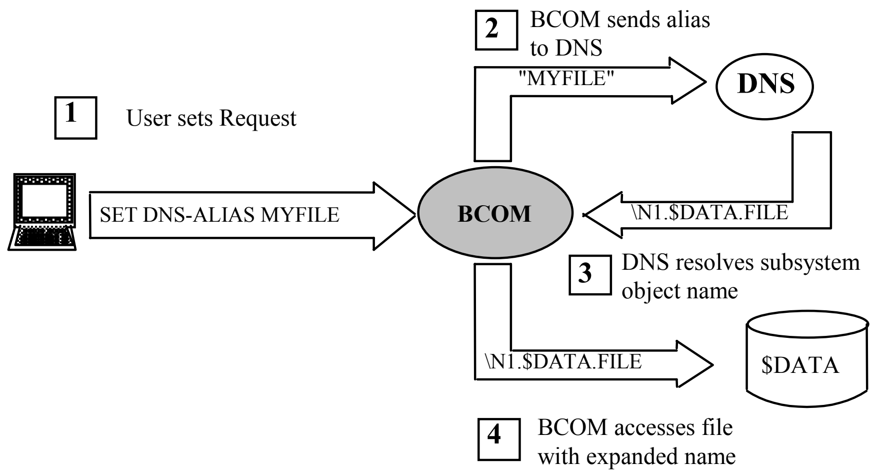
User Exit Routines

BCOM offers exit routines on NSK Guardian. These routines are systematically organized at various key event points that occur during a transfer.

NSK Guardian-Specific FEATURES

DNS Aliases

All file names used on the NSK Guardian node as either the target or source of a file transfer can be referred to by means of Distributed Name Services (DNS) alias names.


1 User sets Request
SET DNS-ALIAS MYFILE
BCOM
2 BCOM sends alias to DNS
"MYFILE"
DNS
\N1.$DATA.FILE
3 DNS resolves subsystem object name
\N1.$DATA.FILE
$DATA
4 BCOM accesses file with expanded name

Event Management System (EMS)

EMS is a collection of processes, tools and interfaces that support the reporting and retrieval of event information. Information retrieved from EMS can help you to monitor your system or network environment, analyze circumstances that led up to a problem, detect failure patterns, adjust for changes in the runtime environment, recognize and handle critical problems and perform many other tasks required to maintain a productive computing operation.

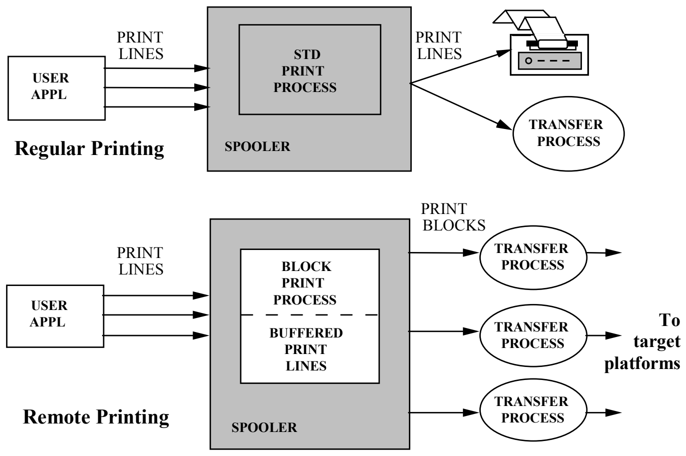
With BCOM, EMS users will be able to choose where the messages generated by BCOM will be routed. You will be able to send them to:

- a log file
- a printer, terminal or spooler location
- $0 the operator process collection area where the messages can be displayed using VIEWPOINT, or captured and analyzed by user-written procedures.

Standard & Block Printing Processes

The block print process supports up to eight concurrent transfer processes, and writes out print lines in 4K blocks.

Because of the blocking, IPC overhead is reduced and throughput can be dramatically improved when using high-speed transmission devices. A special configuration option allows BCOM to use static copies of print modules on the Tandem, to minimize process management overhead.



Supported File Types and formats

All Enscribe file types, UNSTRUCTURED EDIT and NON-EDIT file types, STRUCTURED (key, relative and entry sequenced), SQL tables, Tape files, Spool files, and processes are supported on NSK Guardian. The request definition does not have to indicate which type of file is being transferred; BCOM uses Guardian procedures to determine the file type, and then selects the appropriate access method to use.

Enscribe files and SQL tables are handled record by record, with a NSK Guardian file size limitation of 4072 bytes per record. Unstructured Edit files are handled line by line. Unstructured NON-EDIT files are handled as if each file were a continuous string of characters. The string is unblocked to match the corresponding target file record size (for a maximum of 4096 bytes per record).

If the remote location is an MVS platform, then all VSAM (KSDS, ESDS, RRDS), QSAM, GDG and tape files can be used.

Starting at version V5, all software in the BCOM family support format-2 files (“Big Files”), which was introduced in NSK D46 and G06.

Automatic Restart of Abended NSK Guardian Transfer Process

Once a NSK Guardian transfer process has started up successfully, BCOM will immediately restart it if for some reason it abends.

BCOM options

Backup and Restore to a BCOM server

Backup/Restore is an option that enables the user to back up NSK Guardian files to an MVS, and to restore those files at any time. With this option, you can back up NSK Guardian files either to an MVS sequential file or directly to tape. This allows a NSK Guardian network to make use of the existing MVS tape management systems and procedures.

This option includes a stand-alone BCOM NSK Guardian utility that may be used for disaster recovery. This utility provides a means to restore a BCOM Backup when the NSK Guardian-MVS connection is not available. The MVS tape, which was created by the BCOM backup, can be physically loaded on the NSK Guardian tape drive and then processed by the BCOM Tape utility.

Backup and Restore to a Tivoli Storage Manager server

TSM Backup/Restore is an option that allows storing the output of a NSK Guardian backup in a TSM object stored on a remote Tivoli Storage Manager server, running on MVS or another supported platform.

This option retains on NSK Guardian the regular BCOM request and process management, as in Backup to a BCOM server, but the communication is handled by TSM software. Stand-alone restore is not supported.

MULTIPLEX

This option allows you to define transfer sessions that will be executed using many SNA lines simultaneously: data is divided in parts that are sent over multiple lines, and replaced in order at destination. This option represents very interesting possibilities of reducing telecommunication costs.

BCOM/FTP

With this option, BCOM requests can be defined for transfers that will use TCP/IP lines. BCOM then participates in transfer sessions with a remote FTP application that could be on any type of platform (e.g. UNIX). In addition, as the BCOM/FTP option is a feature of BCOM, a platform with BCOM/FTP can act as a transition node for transfers that could be achieved using both TCP/IP and SNA protocols.

BCOM-EP

With this option, BCOM requests can be defined for transfers that will use TCP/IP lines. This option is different from the pervious BCOM/FTP option: the remote partner for BCOM-EP is another BCOM-EP. Currently, BCOM-EP is available only on IBM/MVS and NSK Guardian.

Increased maximum of LUs

This option allows the use of up to 64 SNA LUs for defining client and server processes. The standard version of BCOM allows for a maximum of 32 connections.

The BCOM Interface

With the BFSINT user interface BCOM users can communicate their transfer needs to BCOM. The BFSINT facility is command driven in nature, which means that users may execute the appropriate commands to define and queue their requests to BCOM. In addition BCOM is designed so that the user does not have to repeatedly enter the same long commands each time he defines and queues a request.

There are three ways to start BFSINT, depending on how you wish to work with it. These execution modes are:

- The Interactive Mode
- The Batch Mode
- The Programmatic Mode
- The Single Command Mode

Execution modes and the commands required to invoke them are described in detail below.

Method 1: Interactive Mode

This method allows you to establish an interactive line mode session with the BFSINT module. This method is similar to an interactive session with TACL, with the exception that the line prompt is the ~ character as opposed to TACL's '>' prompt.

To invoke BFSINT in the interactive mode, issue the following command in response to the TACL prompt:

```
TACL> RUN BFSINT $FSMON
```

Where:

BFSINT	is the name of the BFSINT program object file;
$FSMON	is the Monitor process name chosen at BCOM installation time.

For example, if the monitor process name is MON1; the command would be entered as follows:

```
RUN BFSINT $MON1
```

The default monitor process name is $BMON. If this name was selected at installation time, then the command would be entered as follows:

```
RUN BFSINT
```

Method 2: Batch Mode

This execution mode allows you to batch together a series of BFSINT commands in an edit file and have them processed by BCOM. You can do this in two modes:

1) By INFILE:

To invoke BFSINT in batch mode, issue the following command:

```
RUN BFSINT /IN command-file/ $BMON
```

where:

BFSINT	is the name of the BFSINT program object file
command-file	is the name of the file containing the BFSINT commands;
$BMON	is the name of the monitor process name. If the default monitor process name ($BMON) was selected at installation time, then it does not have to be included in the command. Like this: `RUN BFSINT /IN filexyz/`

2) By startup parameter on the TACL RUN:

```
RUN BFSINT $BMON; OBEY <command-file>
```

where:

$BMON	is the name of the monitor process name. For example:
<command-file>	is the name of the file containing the BFSINT commands;

Example: `RUN BFSINT $BMON; OBEY filexyz`

Method 3: Programmatic Mode

This section describes how to use BFSINT at the program level.

Two startup modes

BFSINT can be started up in either interactive or non-interactive mode.

Interactive mode

When you start up BFSINT in interactive mode, BFSINT will always terminate normally (i.e. by issuing a CALL TO STOP).

Non Interactive mode

When you start up BFSINT in non-interactive mode:

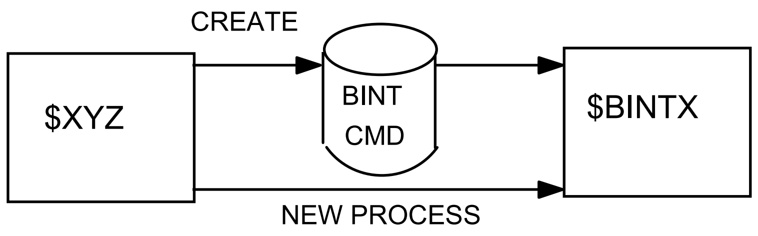
- If it terminates normally, it issues a CALL STOP
- If an error should be encountered, it will terminate by issuing a CALL ABEND.

When an application sends the startup message

In both batch and interactive modes, when an application sends a startup message BFSINT will reply with an error 70 (continue file operation). This message tells the application that the PARAM message can now be sent.

Case 1: Startup in Non Interactive Mode: Send one command to BFSINT

The following example demonstrates how BFSINT is used to submit a BCOM request and check the completion of the transfer:



```
INT BINT^OBJ     [0:11] := ["$VOL    SUBVOL  BFSINT  "];  ! BFSINT object pgm.
INT BINT^PROCID  [0:11] := [12 * ["   "]];                ! BFSINT process^id.
INT BINT^PROCNAME[0:3]  := ["$BINTX"];                    ! BFSINT procname.
INT BINT^FNUM;
INT ERROR;

STRUCT STARTUP (STARTUP^MSG^DEF);

PROC MAIN^PROC MAIN;

BEGIN

  STARTUP.MSGID    :=  -1;
  STARTUP.DEFAULT ':=' ["$VOLUME SUBVOL  "];         ! Volume and subvolume
  STARTUP.IN      ':=' ["$XYZ                    "]; ! Process name of this pgm
  STARTUP.OUT     ':=' ["$XYZ                    "]; ! Process name of this pgm
  STARTUP.PARAM   ':=' ["$BMONX;QUEUE #REQUEST"];    ! Monitor process name and
                                                     ! QUEUE #request command for BINT

!--------------------------------------------------------------------------------!
! Create and open BFSINT process                                                 !
! Send startup message with COMMAND BINT in startup param                        !
!--------------------------------------------------------------------------------!

  CALL NEWPROCESS (BINT^OBJ,,,, PROCID, ERROR, BINT^PROCNAME);
  CALL OPEN  (PROCID, BINT^FNUM, 0, 1);
  CALL WRITE (BINT^FNUM, STARTUP, $LEN (STARTUP));
  CALL CLOSE (BINT^FNUM);                !* After sending startup msg then close BINT

!--------------------------------------------------------------------------------!
! - BFSINT open this process                                                     !
! - Treat and reply the message from BFSINT                                      !
! - When the transfer completes, this process will receive a system message:     !
!    Stop-Message  if the transfer completes successfully or                     !
!    Abend-Message if the transfer completes unsuccessfully.                     !
!--------------------------------------------------------------------------------!
END;   ! PROGRAM
```

The command could be an Obey file of commands. The program ($XYZ) can create an Obey file of commands of BCOM or use an existing file and then create the BFSINT process, pass the Obey file name in parameter of startup message or as the IN file of startup message. The OUT file of startup message can be the program ($XYZ), spooler or terminal.

Examples:

1. Obey file in parameter of startup:

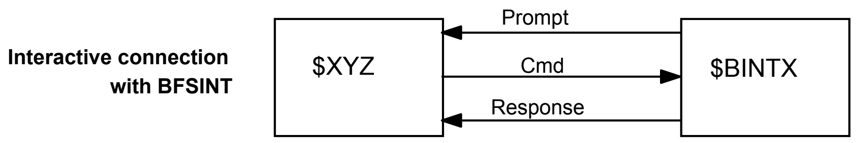
```
STARTUP.DEFAULT ':=' ["$VOLUME SUBVOL  "];         ! Volume and subvolume
STARTUP.IN      ':=' ["$XYZ                    "]; ! Process name of this pgm
STARTUP.OUT     ':=' ["$XYZ                    "]; ! Process name of this pgm
STARTUP.PARAM   ':=' ["$BMONX; OBEY OBEYFILE"];    ! Monitor proc. and Obey file names
```

2. Obey file in Infile of startup:

```
STARTUP.IN      ':=' ["$VOLUME SUBVOL  OBEYFILE"]; ! Process name of this pgm
STARTUP.OUT     ':=' ["$XYZ                    "]; ! Process name of this pgm
STARTUP.PARAM   ':=' ["$BMONX;"];                  ! Monitor process name
```

Remark *You can consult the example-program (under name PGBINTX1) included in the BCOM package.*

<u>Case 2</u>: Startup in Non Interactive Mode: Receive Response from BFSINT



You can also create a BFSINT process, send it a startup message with INFILE and OUTFILE set to your process name ($XYZ). The BFSINT process will then open your process and issue a prompt (message with length = 1). Once the prompt is received, a command can be sent via the Guardian reply.

The following example illustrates this technique:

```
. . .
STARTUP.INFILE  ':=' "$XYZ                    ";
STARTUP.OUTFILE ':=' "$XYZ                    ";
STARTUP.PARAM   ':=' ["$BMONX"]
BINT^OBJ        ':=' "$YOUR   BINT    OBJ     ";
PROCESS^NAME    ':=' "$BINTX ";

CALL NEWPROCESS (BINT^OBJ,,,, PROCESS^ID, ERROR, PROCESS^NAME);

CALL OPEN  (PROCESS ^ID, BINT ^FN);
CALL WRITE (BINT^FN, STARTUP, STARTUP^LEN);
CALL CLOSE (BINT^FN);
WHILE NOT FINISHED DO
  BEGIN
    CALL READUPDATE (RECEIVE^FN, BUFFER, READ^COUNT);
    IF <> THEN
      CALL ERROR^PROCESS;
    CALL RECEIVEINFO (P^NAME,,,,BINT^READ^COUNT);
    IF P^NAME = PROCESS ^NAME FOR 6 BYTEs THEN
      IF BINT^READ^COUNT = 1             ! count = 1, prompt message
        THEN CALL SEND^A^BINT^CMD        ! Yes, BINT ready to rcv cmd
        ELSE CALL ANALYZE^BINT^RESPONSE; ! No, count <> 1, analyze
                                         ! BFSINT response
  . . .
  END;
```

The program should reply only with a zero reply code when the message received is a response. BINT^READ^COUNT <> 1 (not a prompt message).

The BFSINT response has the same format as a conversational terminal response.

When the message is a prompt (BINT^READ^COUNT = 1), the process replies with the commands to be executed or with a reply code 1 to indicate the end.

Note: *In the case of an unsuccessfully QUEUEWAIT request, and the IN and OUT files are the same process, BFSINT will send a CONTROLBUF message (-35) (Ignore this message when it is received).*

Remark: *You can consult the example-program (under name PGBINTX2) included in the BCOM package*

Case 3: Startup Using Interactive Method

Always use the startup message.

```
STRUCT  STARTUP;
   BEGIN
   INT     MSGCODE;        ! -1 = Startup message code
   INT     DEFAULT[0:7];   ! default Volume, Subvolume
   INT     IN^FILE[0:11];  ! In-file name
   INT     OUT^FILE[0:11]  ! Out-file name
   INT     BUF[0:39];      ! Parameter string of the Run command, null
   STRING  S^BUF = BUF;
END;
```

Only use the PARAM-message to override the Interactive setting of BFSINT:

```
PARAM INTERACTIVE YES

STRUCT  PARAM;
   BEGIN
   INT     MSGCODE;        !-3 = PARAM message code
   INT     NUMPARAMS;      !Number of PARAM msgs included in this message
   STRUCT  MSGS;           !Beginning of PARAM messages
      BEGIN
      STRING  P^LEN;             !Length "n" of parameter-name
      STRING  P^NAME[0:N-1]      !Parameter-name for "n" bytes
      STRING  V^LEN;             !Length "v" of parameter-value
      STRING  V^VALUE[0:V-1];    !Parameter-value for "v" bytes
      END;
   END;
```

Sample STARTUP message:

```
STARTUP.MSGCODE  :=  -1;
STARTUP.DEFAULT ':=' "                 ";
STARTUP.INFILE  ':=' "$XYZ                    ";
STARTUP.OUTFILE ':=' "$XYZ                    ";
STARTUP.BUF      :=  0;
```

Sample PARAM message:

```
PARAM.MSGCODE       :=  -3
PARAM.NUMPARAMS     :=  1;
PARAM.MSGS.P^LEN    :=  11;
PARAM.MSGS.P^NAME  ':=' "INTERACTIVE";
PARAM.MSGS.V^LEN    :=  3
PARAM.MSGS.V^VALUE ':=' "YES";

! Create the BINT process :

BINT^PROGRAM^FILE ':=' <BINT-OBJECT-FILE-NAME>    ! $SYSTEM.BCOM.BINT
BINT^PROCESS^NAME ':=' <BINT-PROCESS-NAME>        ! $BINT
CALL NEWPROCESS (BINT^PROGRAM^FILE,,,,PROCESS^ID,ERROR,PROCESS^NAME)
CALL OPEN  (PROCESS^ID, BINT^FNUM);
CALL WRITE (BINT^FNUM, STARTUP,$LEN (STARTUP));   ! Error 70 is normal
                                                  ! after the write
CALL WRITE (BINT^FNUM, PARAM, $LEN (PARAM));
CALL CLOSE (BINT^FNUM);
```

Method 4: Single Command Mode

This execution mode allows you to run a copy of the BFSINT in order to process a single command. The BFSINT stops after having processed the command successfully - or abends, otherwise.

The single command is specified as a STARTUP message parameter on the TACL RUN command:

```
RUN BFSINT $MON1; single-command
```

where:

BFSINT	is the name of the BFSINT program object file
$MON1	is the name of the monitor process name. This name defaults to $BMON
single-command	is the single BFSINT command that will be processed by the program; it is prefixed by a semicolon

Defining BCOM requests

This chapter is organized according to the following sections:

1. Request/Queue Architecture gives an overview of Requests and Queues, and describes the relationship between them.

2. Request Definition Commands provides an overview of the BFSINT command categories and of the commands within those categories.

   This section is meant to familiarize users with the commands they will be entering when setting up and queuing their requests, and when controlling the BCOM environment.

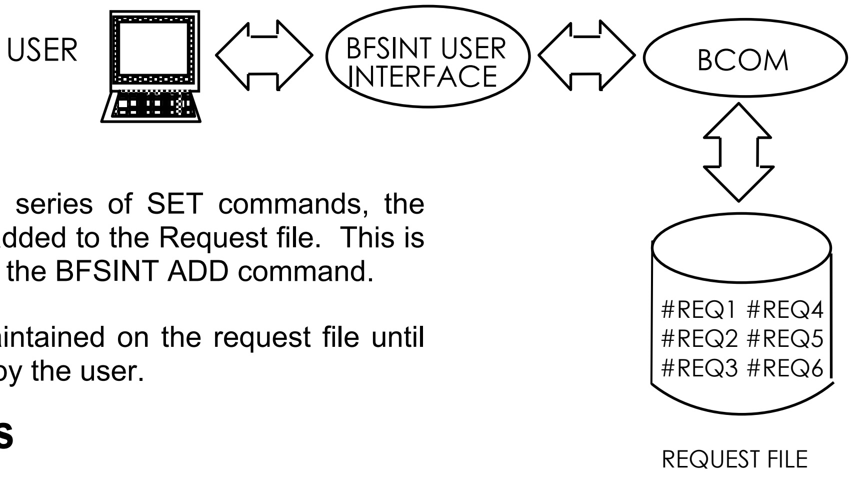
3. Request Control Facilities outlines the operations that can be performed to manage requests from within a BFSINT session. It discusses the commands required to do this and includes examples of how they are used.

4. Queue Control Facilities discusses the operations that can be performed from within a BFSINT session to manage queued requests. It discusses the individual commands needed to do this, and provides examples of how they can be used.

The request/queue architecture

Request basics

Transfer requests are defined by BCOM users through the BFSINT user interface. The definitions of these transfer requests are stored on the BCOM Request file. After all transfer attributes have been described to the BFSINT user

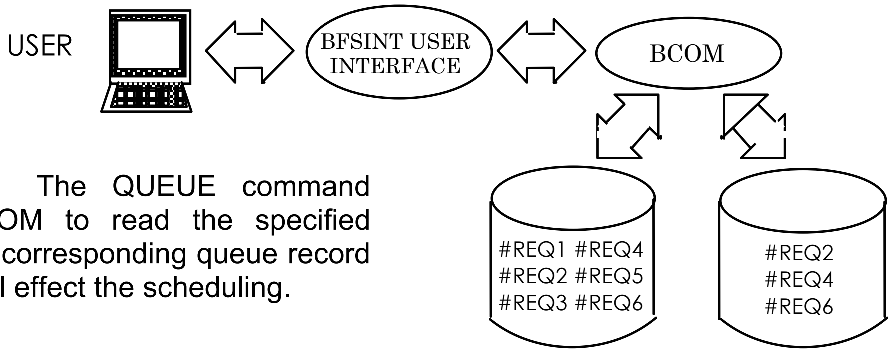


interface using a series of SET commands, the request may be added to the Request file. This is accomplished via the BFSINT ADD command.

Requests are maintained on the request file until explicitly purged by the user.

Queue basics

Once a request has been defined through BFSINT and stored on the Request file, it is ready to be queued for execution. This is accomplished via the BFSINT



QUEUE command. The QUEUE command actually causes BCOM to read the specified request and create a corresponding queue record from which BCOM will effect the scheduling.

Note: *The entry in the Queue file pertaining to a given request is always deleted at the end of the transfer job regardless of whether or not the transfer was successful.*

The outcome of the transfer can be ascertained by means of a BFSINT STATUS request-name command. See the subsection entitled Request Control Facilities later in this section for a further discussion of the STATUS, LISTREQ, INFO, PURGE, QDEL, QHOLD, QLIST, QREL, and RESET commands.

This subsection provides an introduction to the different aspects of Request Definition processing using the BFSINT commands. Detailed syntax conventions associated with each command are not provided here. For an in-depth explanation of individual command syntax conventions, please consult *Appendix - Command Syntax and Reference Summary.*

BFSINT request definition commands

The BFSINT Request Definition Commands are the ADD, SET, RESET and SHOW commands.

These four commands are used to define a transfer request to BCOM before actual queuing and execution of the request. Once request definition commands have been issued, the request is stored on the BCOM local Request file and is ready for queuing or execution.

The individual Request Definition Commands are described below.

> ***Note:*** *For correct individual command syntax, please refer to* Appendix - Command Syntax and Reference Summary *for more details.*

The ADD Command

This command physically creates a request entry on the BCOM Request file, which may subsequently be queued for execution. All request names must be at most 8 characters in length and must begin with a "#".

The SET Command

This command, as the name implies, sets the information and attributes of the individual request.

The SET command captures information such as: local location, remote location, request type, local file information, remote file information, request options, etc… These are:

SET TYPE FB

Indicate that this is a file backup request using the specialized BACKUP/RESTORE optional feature of BCOM.

SET TYPE FR

Indicate that this is a file restore request using the specialized BACKUP/RESTORE optional feature of BCOM.

SET TYPE SF|RF

Indicate that this is a file transfer type request: SF (Send File) is an outbound transfer request and RF (Receive File) is an inbound transfer request.

SET TYPE LJ|RJ

Indicate that this is a job submission request (e.g., MVS JCL, NSK Guardian TACL, etc.). If the job executed locally, then the type is LJ (Local-job) otherwise the type is RJ (Remote-job).

SET TYPE RQ

Indicate that this is a request for BCOM to queue a request definition at a remote location.

SET LOCAL-LOCATION logical-location-name

Indicate the location of the current BFSMON process. The name begins with a % sign and must exist in the configuration file.

SET REMOTE-LOCATION logical-location-name

Indicate the location of the target node to which files are transferred and from which files are received. The remote file name must be defined in the configuration file and begin with a % sign.

SET FIELD field-offset, length [, field-offset, length, ...]

This command allows users to bypass the ASCII/EBCDIC conversion, which normally takes place for file transfers between ASCII-based and EBCDIC-based platforms. This translation bypass effectively allows users to accomplish correct handling of binary data fields between the two communicating partners.

SET ROUTING-CLASS routing-class-value

The ROUTING-CLASS governs the physical transmission and service properties over which the file transfer will take place. Usually, ROUTING-CLASS is reserved for particular groups of LUs, which are defined over differing line speed capabilities. For more information refer to the previous chapter for further details on ROUTING-CLASS.

SET PRIORITY-CLASS numeric-value

Indicate what scheduling priority the request should have in the Queue file. This will enable it to be scheduled for execution.

SET CHECKPOINT checkpoint-value

For large file transfers over lower speed lines, the CHECKPOINTing feature can be used to minimize the number of records that must be retransmitted if the initial transfer request fails. BCOM will keep track of the last record transferred every "checkpoint-value" records. Thus, in the event of an abend the request may be restarted at the last successfully checkpointed record in both the source and destination files.

SET LOCAL-FILE file-name

Indicates the file name on the LOCAL location node where the file participating in this transfer resides. Network file names are allowed.

SET LOCAL-ATTRnn (Range of nn: 00 - 10)

Define the attributes of the local-file, as required by the type of transfer request defined. (for FTP ATTRIBUTES, see the *BCOM/FTP on NSK Guardian Guide*).

(For TSM ATTRIBUTES, see the *BackHome/TSM Guide*)

Examples of LOCAL-FILE-ATTRnn values on NSK Guardian platforms are: OUTFILE, PROCESS-PRIORITY, BRPARAMS, PARAMS etc… File attributes are node specific. The particular file attribute designator is placed as the first operand of the user portion of the command, and is followed by the attribute's actual value.

SET LOCAL-ATTRnn PROCESS-PRIORITY=priority-value

Indicate the priority of the NSK Guardian I/O process that will be accomplishing the I/O to the target file or process. For a description of the software modules employed for each file-type, consult the BCOM on NSK Guardian Installation & Operations Guide.

This priority overrides any value configured by IO-PROCESS-PRI for the transfer process running the request. Note that for local-job requests (LJ) initiated on NSK Guardian platform, there is no transfer process involved and the default value is the process priority of the monitor minus 1.

SET LOCAL-ATTRnn PARAMS=utility-parameter-values

Indicate utility specific parameter options used by NSK Guardian utilities that are being used to accomplish the transfer I/O. For file transfer in which the type is either SF/RJ/LJ, share may be specified.

SET LOCAL-ATTRnn BRPARAMS=backup/restore-option

Indicate the fileset and any option required for the Backup/Restore.

Only valid for file transfer in which the type is either FB or FR.

SET LOCAL-ATTRnn OUTFILE=NSK Guardian-file-name

Indicates the NSK Guardian OUT file to which RUN TIME output for file transfers whose types are FB, FR, LJ, should be directed.

SET LOCAL-ATTRnn REPORT {ON | OFF}

Indicate whether the first character of every record on the MVS FILE should be interpreted as a valid print control character (ON). Valid only for MVS locations with print datasets.

SET REMOTE-FILE file-name

Indicate the file name on the REMOTE-LOCATION node where the file participating in this transfer resides. Also use to specify the name of a remote request to be queued at the remote location.

SET REMOTE-ATTRnn

Define the attributes of the remote file, as required for the type of request defined. The nn is a consecutive number that starts at 01 and increases for each attribute to be defined. (for FTP ATTRIBUTES, see the BCOM/FTP on NSK Guardian Guide).

(For TSM ATTRIBUTES, see the BackHome/TSM Guide)

Examples of REMOTE-ATTRnn values for a remote MVS are: DCB, DISP, LABEL, UNIT, PARMS, SPACE, VOL, etc… File attributes are node specific. The particular file attribute designator is placed as the first operand of the user portion of the command, and is followed by the attribute's actual value. Please refer to the SET command in description in the *Appendix - Command Syntax and Reference Summary*.

BFSINT Session State Commands

Two additional commands allow the user to control and manage the request as it is being prepared. All of the SET commands described in the previous section are maintained internally for each session, and can be described as the BFSINT Session State.

BFSINT Session States are similar in function to corresponding concepts used by the Pathway subsystem in a NSK Guardian environment. The session states guarantee that current values for all SET commands will be "remembered" by the BFSINT interface. Thus, you can SET and RESET BFSINT commands numerous times until the request information is in the state you want it to be in. The request can then be added to the request file.

Note: *For correct individual command syntax, please refer consult* Appendix - Command Syntax and Reference Summary*.*

The RESET Command

This command allows the user to reset individual attributes for the current BFSINT Session State to their default values. This initialization only applies to the Session State and not to the Request file.

When the BFSINT process has been started with certain default values for Routing Class, remote location and local location, these values are substituted automatically into the context area in preparation for the next request definition.

The SHOW Command

This command will show the current values for all predefined information items and attributes. With SHOW, the user can verify all the BFSINT SET values that have been entered.

The SHOW command should be used before adding a request, to insure that all the attributes are set correctly.

The SHOW command, used without any parameter, displays the current request attributes for this BFSINT session. When a BFSINT session is started, some attributes are initialized to their default values, and the remaining ones are left undefined. To change their values, use the SET or the RESET command. The SHOW command displays the current attribute values that will be used in the event an ADD request-name command is executed.

Request Control Facilities

Now that requests have been added to the Request file, all this information can now be displayed, changed, and/or managed using the commands described below.

Request control facilities include commands that allow you to:

- Display request attributes (INFO)
- Display a list of requests (LISTREQ)
- Delete a request from the request file (PURGE)
- Set attributes to match those of an existing request (SET LIKE)
- Display the setting of a current request (SHOW).

All of these operations are described below. Actual examples of how requests might be managed and controlled are then presented.

Note: *For correct individual command syntax, please refer to* Appendix - Command Syntax and Reference Summary*.*

The INFO Command

This command is used to display all the information and request attributes for one or more requests in the Request file.

The LISTREQ Command

This command displays the contents of the Request file so that the user can view previously added requests. The requests may be displayed generically, by leading characters. The command will simply list the request names.

The command displays the alphabetical list of all or a subset of the Request file. A subset is determined by specifying either a generic name (the first character(s) of a request name), or the entire request name.

If a generic name is specified, all the requests whose first characters match this generic name will be listed.

The PURGE Command

When a request is no longer used, it can be deleted from the Request file using the PURGE command. After this is done, it is possible to add another request with the same name to the file.

The SET LIKE Command

The SET LIKE command is used to match new request attributes to the attributes of an existing request. The attributes of the request specified in this command become part of the session state's current attributes. The SET and RESET commands can then be used as usual, before adding the new request.

Request Management Examples

Display File Transfer Attributes

To display file transfer attributes for the #EXAMP21 request:

```
~INFO  #EXAMP21
  REQUEST            = #EXAMP21
  TYPE               = RECEIVE-FILE
  COMMENTS:
    01 THIS IS THE FIRST COMMENT LINE
    02 THIS IS THE SECOND COMMENT LINE
    03 THIS IS THE THIRD COMMENT LINE, ETC...
  OWNERID            = 130,50
  ROUTING-CLASS      = 19
  PRIORITY-CLASS     = 5
  LOCAL-LOCATION     = %Q1T
  LOCAL-FILE         = $DATA.SUBVOL.TANFILE
  LOCAL-ATTRIBUTES:
    01 PROCESS-PRIORITY=150
  REMOTE-LOCATION    = %Q1M
  REMOTE-FILE        = TEST.DOWNLOAD.DATA
  REMOTE-ATTRIBUTES:
    01 DISP=SHR
  AUTO-RESTART       = NO
  CHECKPOINT (REC.)  = 15000
  COMPRESSION        = NO
  NORMAL-END         = <None>
  ABNORMAL-END       = <None>
  FIELD              = NONE
```

Display Request Names

To display all request names:

```
~ LISTREQ
#EXAMP10 #EXAMP11 #EXAMP12 #EXAMP13 #EXAMP14 #EXAMP20 #EXAMP21
```

To display all the request names beginning with #EXAMP1:

```
~ LISTREQ #EXAMP1
#EXAMP10  #EXAMP11  #EXAMP12  #EXAMP13  #EXAMP14
```

Delete a Request

To delete a request from the Request file:

```
PURGE  #EXAMP10
#EXAMP10  PURGED
```

Match Attributes of New Request to Existing Request

A new request definition can be made using an existing request as a skeleton or model. To set the current attributes to an existing request, use the SET LIKE command:

```
SET  LIKE  #EXAMP21
```

The request definition attributes are now set to those of #EXAMP21. The user may now alter or tailor some of these in order to create the specific parameters of a new request - such as change the routing class to 07.

Display Request Settings

To display the current request definition settings, use the SHOW command:

```
~SHOW

REQUEST PARAMETERS
  REQUEST-TYPE        = RECEIVE-FILE
  COMMENTS:
    01 THIS IS THE FIRST COMMENT LINE
    02 THIS IS THE SECOND COMMENT LINE
    03 THIS IS THE THIRD COMMENT LINE, ETC...
  ROUTING-CLASS       = 7
  PRIORITY-CLASS      = 5
  LOCAL-LOCATION      = %Q1T
  LOCAL-FILE          = $DATA.SUBVOL.TANFILE
  LOCAL-FILE-ATTRIBUTES:
    01    PROCESS-PRIORITY=150
  REMOTE-LOCATION     = %Q1M
  REMOTE-FILE         = TEST.DOWNLOAD.DATA
  REMOTE-FILE-ATTRIBUTES:
    01    DISP=SHR
  AUTO-RESTART        = NO
  CHECKPOINT          = 1500
  COMPRESSION         = NO
  ABNORMAL-END        = <None>
  NORMAL-END          = <None>
```

In this example, we can see the modified routing class (07) on request attributes copied from those of #EXAMP21 through the previous SET LIKE command.

Queue Control Facilities

The QUEUE commands are used to define the queue option overrides and to initiate a file transfer. Additional operations can be performed to initiate or control the execution of a file transfer. For instance, you can:

- Schedule a request for transfer (QUEUE)
- Abort a transfer (ABORT)
- Prevent a queued request from being executed (QUEUEHOLD, QUEUEDEL)
- Display the list of queued requests (QUEUELIST)
- Release a held request (QUEUEREL)
- Queue interactively (QUEUEWAIT)
- Display the status of queued requests (STATUS).

The QUEUE Command

This command physically places a request that has been previously defined through BFSINT onto the execution queue for scheduling. The moment the request is processed by the user interface, BFSINT will prompt for the next command.

The ROUTING-CLASS and PRIORITY-CLASS can also be specified when a request is queued. When restarting a previously abended request which was running with CHECKPOINT, the user may specify RESTART when reQUEUEing the request.

Refer to the QUEUEWAIT command for an example of another type of queuing.

The ABORT Command

This command can be used to stop a request that is currently executing from completing normally, or to interrupt a transfer process, or even all transfer processes. The ABORT command can also be used to abort specific transfer processes. Refer to the Operations section of the *Appendix - Command Syntax and Reference Summary* for details on the ABORT facility.

> ***Note:*** *It is recommended to avoid using the ABORT command to stop an executing file transfer request, whenever possible, since the destination file may be left in a corrupted state.*

QUEUEHOLD, QUEUEDEL and QUEUEREL Commands

To prevent activation of a queued request, either by temporarily holding the request or removing it from the queue, use the QUEUEHOLD and QUEUEDEL commands. QUEUEHOLD holds the request and QUEUEDEL removes it from the queue (not from Request file). A request that has been held does not lose its rank in the queue. It can be released at any time with the QUEUEREL command.

QUEUELIST Command

This command simply lists the requests that currently reside on the Queue file scheduled for execution, as well as the requests that are actually executing. Additional information related to the specifics of each queued request is also displayed with this command.

The QUEUELIST; command can be used to display the list of the requests currently on the Queue file. This list shows the request names, the date and time they were added to the queue, their status (Waiting, Held or Executing) and their queue options.

The QUEUEWAIT Command

The QUEUEWAIT command allows your BFSINT process to queue a request and wait for its completion before returning to your next command. This is useful if you wish to include the queuing of a request within a BCOM Obey file, where the remaining commands must not be executed prior to completion of the transfer.

The STATUS Command

The STATUS command provides detailed information on the state of either an individual request, or of a transfer process within a particular Routing Class. In the case of an individual request, data on the state of the currently executing request will be displayed. If no request is currently executing, the information displayed will relate to the last request executed with the specified #request-name.

In the case of a particular Routing Class, information on the current operational aspects of the specified process name will be displayed. This information contains items such as LU session information, requests queued for this process, number of blocks transferred, connection status, etc.

As soon as a request has been queued at least once, you can get information on the status of its execution by using the STATUS command. This information consists of the date and time queued, started and ended, the number of records and blocks transferred, the status of the request (i.e. EXECUTING, SELECTED, HOLD STATE, ABORTING, TRANSFER SUCCESSFUL, etc.), and a comment. If the request is queued but not yet started, only the queue date and time are displayed. If it is currently executing, only the end date and time are not displayed.

The counts reflect the situation at the time the status command was entered. If the request is not queued, information about the last execution is displayed. This information is kept in the Request file.

Queue Management Examples

Aborting an Executing File Transfer

```
~ABORT  #EXAMP1
REQUEST #EXAMP1 HAS BEEN ABORTED
```

Deleting a Request from the Queue

```
~ QUEUEDEL  #EXAMP1
#EXAMP1  DELETED
~
```

Holding and Releasing a Request

```
~QUEUEHOLD  #EXAMP1
#EXAMP1  HAS  BEEN  HELD

~QLIST

REQUEST  PRI TP QTIME                STATUS    RLOC     RTC  CHPT  RST  OWNER

#EXAMP1  5   FB 1998-10-10 10:15:06  HELD      %Q1D     1    NO    NO   100,50
#EXAMP2  4   FB 1998-10-10 10:21:13  WAITING   %Q1D2    2    NO    NO   100,50

~QUEUEREL  #EXAMP1
REQUEST  #EXAMP1  HAS  BEEN  RELEASED

~QLIST

REQUEST  PRI TP QTIME                STATUS    RLOC     RTC  CHPT  RST  OWNER

#EXAMP1  5   FB 1998-10-10 10:15:06  EXECUTING %Q1D     1    NO    NO   100,50
#EXAMP2  4   FB 1998-10-10 10:21:13  WAITING   %Q1D2    2    NO    NO   100,50
```

Queue Wait a Request

```
~ QUEUEWAIT  #EXAMP1
#EXAMP1  FILE TRANSFER HAS BEEN INITIATED...
#EXAMP1  HAS COMPLETED SUCCESSFULLY
REQUEST NAME = #EXAMP1
QUEUE TIME = 1998-03-31  13:16:57
START TIME = 1998-03-31  13:17:05
END TIME = 1998-03-31  13:17:16
RECORD COUNT = 3
BLOCK COUNT = 1
MESSAGE = TRANSFER  SUCCESSFUL
```

Display the Execution Status of a Request

```
~STATUS #EXAMPL1

*** CURRENT STATUS AT 1998-03-21  13:19:49

REQUEST NAME         = #EXAMPL1
TYPE                 = SEND-FILE
QUEUE TIME           = 1998-03-31 13:16:57
START TIME           = 1998-03-31 13:17:05
END TIME             = 1998-03-31 13:17:16
QUEUE USER ID        = 130,255
RECORD COUNT         = 3
BLOCK COUNT          = 1
BFSBKUP  COMPL. CODE = 0
BFSBKUP  TERM.       = <None>
JOB-ID               = $X3Z6
JOB-OBJ              = FUP
MESSAGE              = TRANSFER SUCCESSFUL
```

This is the primary means of establishing the outcome of the completed transfer. The following are possible values that may be received in the MESSAGE area:

```
TRANSFER SUCCESSFUL
ABNORMAL TERMINATION
EXECUTING
SELECTED
FILE ALLOCATION ERROR
WAITING
OPEN ERROR
TRANSFER ERROR
DELETED BY OPERATOR
HOLD STATE
ABORTING
AWAITING COMPLETION
HOLD FOR RETRY
```

BCOM file transfers

This chapter explains how to execute BCOM Requests. As such, it illustrates the basic categories of transfer requests supported by the product and provides several examples of how to define, queue and control the execution of these requests.

The chapter is organized according to the following sections:

1. Procedures for Defining File Transfer Requests explain how to send files from NSK Guardian platforms and how to receive files from remote locations.
2. Data Conversion Utilities explains how to perform selectable field data translation.
3. CheckPoint/ReStart Facilities explains how to associate and use checkpointing information on a file transfer request.
4. Using a process to send blocked data and Using a process to receive blocked data explains how to build blocks and records when BCOM is to use a NSK Guardian process either as source or destination of data.

Defining file transfer requests

In most cases, the transfer request procedure consists of these two steps:

Step I: Define the Request

A) Gather information

Means gathering all the necessary information about the source and destination objects that will be involved in the transfer (file name, dataset name, etc.), then;

B) Set the request attributes

Using the BCOM Interface program (BFSINT), enter the request definition parameters according to the source and destination file characteristics. Then;

C) Add the request

Within the same BFSINT session, add the request to the Request file. Once this is done, the request will be available for queuing any number of times, until it is specifically purged.

Step II: Activate the Request

A) Identify the queue overrides

This step is optional.

B) Queue the request

A request is activated by adding it to the BCOM Queue file; this is known as "queuing" a request. The queued requests are normally processed in a First In First Out (FIFO) sequence within the PRIORITY-CLASS.

The BCOM Interface also provides commands to display request information and to manage the request entries and the queue entries.

Defining NSK Guardian to MVS File Transfer Requests

Steps I and II below provide an outline of how to define NSK Guardian to MVS dataset transfer request. After this outline, two examples, which illustrate these steps, are provided. This example assumes that the BFSINT process through which the file transfer is defined, currently executes at NSK Guardian; a Send File will then be done.

Step I: Define the Request

A file transfer request needs to be defined only the first time it is used. Once it is stored in the BCOM Request file, it remains available until it is specifically purged. If the file transfer you want to execute is already defined, you do not have to go through Step I.

A) Gathering information

I. Gather information on NSK Guardian site:

The Local file: The local file on the NSK Guardian can be a disk file, a tape file, a DNS ALIAS name, or a named process.

- For a disk file, the complete file name (volume/subvolume/file name) must be known (i.e. preallocated).
- For a tape file, the Tape unit to be used must be known.
- For a DNS ALIAS name, the name must be present in the distributed database of names.
- If a named process is to act as the remote file, its process name must be known. The remote file or DNS ALIAS or process does not need to exist at this point.

I/O options: When the local file is an ENSCRIBE disk file, the option SHARE may be used. For non-ENSCRIBE files, the option APPEND may be used. The default priority for BCOM I/O reading process may also be altered in the request definition.

2. Gather information on MVS site:

The REMOTE file: The REMOTE file on MVS can be a disk file or a tape file. It is recommended that this file be created before the transfer is activated. However, BCOM allows you to specify creation parameters such as Data Control Block (disk and tape) and Space (disk only). The following table describes the parameters that should be defined for an existing disk or tape file:

MVS attrs for transfers to an existing dataset	**for disk**	**for tape**
DCB (Data Control Block)	Optional	Optional
DISP (Disposition)	Mandatory	Mandatory
LABEL	not applic.	Optional
SPACE	Optional	not applic.
UNIT	Optional	Optional
VOL (Volume)	Optional	Optional

The MVS logical location name must be known. This name is specified in the BCOM Configuration File and is specified here through the use of the SET LOCAL-LOCATION command.

3. **Gather transfer information:**

   Automatic restart: In case of transmission failure or session error, the request is automatically restarted. If a checkpoint value is set for the request, only the records coming after the checkpoint will be retransmitted.

   Checkpoint: BCOM keeps track of the last record transmitted after each sending of checkpoint-value records. In case of abnormal interruption of the transmission, only the records remaining after this last record will be sent when the session can restart.

   Compression: If enabled when BCOM is configured, the compression feature can be set to ON for the request.

   Field conversion: BCOM converts the data from ASCII to EBCDIC when transferring records to MVS. It is important to know the position and length of the binary fields where conversion override is required.

   Routing class: As many physical paths can be defined for a same remote location when configuring BCOM, the routing class indicates which class of connections can be used to execute the request.

4. **Gather request-chaining information:**

   [Ab]normal end: BCOM can queue another request after the request is completed. Which request is to be queued depends on the first request completion status.

B) Set the request attributes

Using the information gathered as explained above, enter the request definition using the BCOM interface (BFSINT). SET commands for a typical SEND FILE request are:

```
SET TYPE  SF         /* indicates that type is SEND FILE */
SET LOCAL-LOCATION   logical-location-name
SET LOCAL-FILE       NSK Guardian-file-name
SET LOCAL-ATTRNn     attributes
SET PRIORITY-CLASS   priority-value
SET REMOTE-LOCATION  logical-location-name
SET REMOTE-FILE      MVS-file-name
SET REMOTE-ATTRNn    attributes
SET ROUTING-CLASS    routing-class-value
SET CHECKPOINT       checkpoint-count
SET FIELD            offset,length [, offset,length... ] | ALL
SET AUTO-RESTART     YES | NO
SET COMPRESSION      YES | NO
SET COMMENTn         nth text-comments-line
SET NORMAL-END       #request-name
SET ABNORMAL-END     #request-name
```

Notes: *For further information on the SET (AB)NORMAL-END command and its usage refer to the Request Control Facilities section of the previous chapter.*

For further information on the SET CHECKPOINT command and its usage refer to the CheckPoint/Restart Facilities section of this chapter.

For further information on the SET FIELD command and its usage refer to the Data Conversion Utility section of this chapter.

C) Add the request

Immediately following the SET commands (within the same BFSINT session), enter the ADD command with the request name:

```
ADD  #request-name
```

When this command is issued, the request is added to the BCOM Request File with all the attributes that were previously set. It is good practice to use the SHOW command to verify these attributes before adding the request.

Step II: Activate the Request

The request may be activated at any time, within the same BFSINT session or at a later time. In this case, the local file must be present on NSK Guardian. If it is a process, it must be running at the time the request is queued.

A) Identify the queue overrides

This step is optional; for example, the user may wish to:

- RESTART a request which has been run with the CHECKPOINT option and has abended.
- Alter the PRIORITY-CLASS of a request which has already been defined on the Request file and change its PRIORITY-CLASS value.
- Alter the ROUTING-CLASS of a request that has already been defined on the Request file and change its ROUTING-CLASS value.

B) Queue the Request

Following the identification of the optional QUEUE override parameters, enter this command:

```
QUEUE  #request-name  (routing-class, priority-class]
```

BCOM stores this request in the Queue file. There is only one queue for each BCOM environment, so any previously queued request will be processed first. In addition, if a specific destination was selected, your request will be processed only when this process becomes available.

NSK Guardian to MVS - Example 1

File transfer from an NSK Guardian disk file to an MVS disk file through a Send File command entered through the NSK Guardian BFSINT.

Step I: Define the Request

A) Gathering information

1. **Gather information on NSK Guardian site:**
   - The local file on NSK Guardian is a key-sequenced file named $DATA01.MYSUB.DATA.
   - IO options: The local file is to be opened with shared access. We let BCOM assign the priority to the NSK Guardian I/O process.
2. **Gather information on MVS site:**
   - The remote file name is EXAMP1.DSN.
3. **Gather transfer information:**
   - The ROUTING-CLASS used for the transfer will be 01.
   - Checkpointing is to be performed every 50 records.
4. **Gather request chaining information:**
   - In case of normal end, request #REQDNO2 will be automatically queued.

B) Set the request attributes

```
>RUN BFSINT $FSMON
~SET TYPE            SF
~SET LOCAL-LOCATION  %TAND1
~SET LOCAL-FILE      $DATA01.MYSUB.DATA
~SET LOCAL-ATTR01    PARAMS=SHARE
~SET ROUTING-CLASS   1
~SET REMOTE-LOCATION %MVSMTL
~SET REMOTE-FILE     EXAMP1.DSN
~SET REMOTE-ATTR01   DISP=SHR
~SET CHECKPOINT      50
~SET NORMAL-END      #RQDNO2
```

C) Add the request

```
~ ADD  #REQDN01
```

Step II: Activate the Request

A) Identify the queue overrides

There are no specific overrides in this example.

B) Queue the Request

```
~ QUEUE  #REQDN01
```

NSK Guardian to MVS - Example 2

File transfer from an NSK Guardian process to an MVS tape file Once again, the command is entered from the NSK Guardian BFSINT.

Step I: Define the Request

A) Gathering information

1. **Gather information on NSK Guardian site:**
   - The local file is a process named $PROG.
   - IO options: The BCOM I/O process priority must be 170.
2. **Gather information on MVS site:**
   - The remote file name is "EXAMPLE.NO1."
   - The disposition of this file is new so this dataset name must not exist. If the transfer is successful, this file will be catalogued. However, if the transfer is not successful, the file will be deleted.
   - The file is stored as the first dataset sequence number on the standard labelled tape (SL).
   - The UNIT name is TAPE, the volume will be Z10001.
3. **Gather transfer information:**
   - The ROUTING-CLASS required is 03.
   - The PRIORITY-CLASS that the queued request will have is 2.
   - Default ASCII to EBCDIC conversion.
4. **Gather request-chaining information:**

   None

B) Set the request attributes

```
>RUN BFSINT $FSMON
~SET TYPE            SF
~SET LOCAL-LOCATION  %TAND1
~SET LOCAL-FILE      $PROG
~SET LOCAL-ATTR01    PROCESS-PRIORITY=170
~SET REMOTE-LOCATION %MVSMTL
~SET REMOTE-FILE     EXAMPLE.NO1
~SET REMOTE-ATTR01   DCB=(LRECL=133,BLKSIZE=13300,RECFM=FB)
~SET REMOTE-ATTR02   DISP=(NEW,CATLG,DELETE)
~SET REMOTE-ATTR03   UNIT=TAPE
~SET REMOTE-ATTR04   LABEL=(1, SL)
~SET REMOTE-ATTR05   VOL=(Z10001)
~SET ROUTING-CLASS   3
~SET PRIORITY-CLASS  2
```

C) Add the request

```
~ ADD  #REQDN02
```

Notes: *After the REMOTE-ATTRnn designator, syntax is identical to MVS JCL conventions.*

The order of the ATTRnn designators is not prescribed. Any order will be accepted.

<u>Step II: Activate the Request</u>

A) Identify the queue overrides

None for this example.

B) Queue the request

```
~ QUEUE  #REQDN02
```

Defining MVS to NSK Guardian file transfer requests

Steps I and II below provide an outline of how to define an MVS to NSK Guardian file transfer request initiated from the NSK Guardian platform (e.g. a Receive-File with the NSK Guardian platform as the LOCAL location and the MVS platform as the REMOTE location). After this outline, two examples applying these steps are provided. If the file to be transferred contains print control characters, and will be printed on NSK Guardian, refer to the section on Receiving Print files from Remote locations in the NSK Guardian User Guide for specific information.

Step I: Define the Request

A file transfer request only has to be defined the first time it is used. Once it is stored in the BCOM Request file, it remains available for reuse until it is specifically purged. If the file transfer you want to execute is already defined, you do not have to go through Step I.

A) Gathering information

To define a request that involves a file transfer from MVS to NSK Guardian, gather the following information:

1. **Gather information on NSK Guardian site**

   The local file: The local file on NSK Guardian can be a disk file, a tape file referred to by a DNS ALIAS name, or a named process.

   - For a disk file, the complete file name (volume/subvolume/ file name) must be known.
   - For a tape file, the Tape unit to be used must be known, or a DEFINE may be used.
   - DNS ALIAS names are supported. The names must be present in the distributed database of names.
   - Load options: when the local file is an Enscribe disk file, FUP is used as the I/O process. The options described in the NSK Guardian FUP manual for the LOAD command are applicable for a BCOM transfer. See the SET LOCAL-ATTRnn PARAMS attribute. Using BFSLOAD, APPEND is also valid.
   - If a named process is to act as the destination file, its process name must be known. Refer to the section on "Programmatic Interfaces" for further details on NSK Guardian process support.

I/O process priority: BCOM assigns a default priority to the I/O module that will be responsible for writing to the destination file. To alter this priority for a specific request, use the SET LOCAL-ATTRn PROCESS-PRIORITY command.

The logical identifier for the MVS remote location must be known. This value is specified in the BCOM Configuration File. This logical name is specified through the SET REMOTE-LOCATION command.

2. Gather information on MVS site

The remote file: the remote file on MVS can be a disk file or a tape file. The following table describes the parameters that can be defined for the local disk or tape file:

MVS from-attrs for transfers from an existing dataset	for disk	for tape
Data Control Block (DCB)	Optional	Optional
Disposition (DISP)	Mandatory	Mandatory
LABEL	not applic.	Optional*
Space	Optional	
Unit	Optional	Optional
Volume (VOL)	Optional	Optional

* At least one of Dataset Name / Label must be known

Record length: The logical record length (LRECL) should be 4096 or less. If the record format is variable (RECFM = V or RECFM = VB) and the LRECL is greater than 4096, the file transfer will abend (with abend code 3060).

If you are transferring MVS data into an EDIT type file on the NSK Guardian, the FUP will be used to copy the records. Please note that FUP limits the record size of EDIT type files to 132 bytes. Thus, if the size of the MVS record is larger than 132, FUP will create additional records, of up to 132 bytes each, with the remaining length.

3. **Gather transfer information:**

   Automatic restart: In case of transmission failure or session error, the request is automatically restarted. If a checkpoint value is set for the request, only the records coming after the checkpoint will be retransmitted.

   Checkpoint: BCOM keeps track of the last record transmitted after each sending of checkpoint-value records. In case of abnormal interruption of the transmission, only the records remaining after this last record will be sent when the session can restart.

   Compression: If enabled when BCOM is configured, the compression feature can be set to ON for the request.

   Field conversion: BCOM converts the data from ASCII to EBCDIC when transferring records to MVS. It is important to know the position and length of the binary fields where conversion override is required.

   Routing class: As many physical paths can be defined for a same remote location when configuring BCOM, the routing class indicates which class of connections can be used to execute the request.

4. **Gather request-chaining information**

   BCOM allows you to specify the next request name to be executed when this transfer is completed. The subsequent request can be of any type. Different requests may also be chosen depending on the completion status (normal or abnormal). Refer to the Automatic Follow-on Scheduling section for more information.

B) Set the request attributes

Using the information gathered through Step 1, above, you can now enter the request definition using the BCOM Interface program (BFSINT). The following SET commands are used to define the request's attributes:

```
SET TYPE  RF         /* indicates that type is RECEIVE-FILE */
SET LOCAL-LOCATION   logical-location-name
SET LOCAL-FILE       NSK Guardian file name
SET LOCAL-ATTRIBn    attributes
SET REMOTE-LOCATION  logical-location-name
SET REMOTE-FILE      MVS file name
SET REMOTE-ATTRIBn   attributes
SET ROUTING-CLASS    routing-class-value
SET PRIORITY-CLASS   priority-value
SET CHECKPOINT       checkpoint-count
SET NORMAL-END       #request-name
SET ABNORMAL-END     #request-name
SET COMMENTn         text-comments
```

C) Add the request

Immediately following the SET commands (within the same BFSINT session), enter the ADD command with the request name:

```
ADD  request name
```

When this command is issued, the request is added with all the attributes that were previously set. It is good practice to use the SHOW command to verify these attributes before adding the request.

Step II: Activate the Request

The request may be activated at any time, within the same BFSINT session or at a later time. The destination file must be present on NSK Guardian. If a NSK Guardian process is receiving the data for the transfer, it must be running at the time the request is queued.

A) Identify the queue overrides

This step is optional; for example, the user may wish to :

1. RESTART a request that has been run with the CHECKPOINT option and has abended.
2. Alter the PRIORITY-CLASS of a request that has already been defined on the Request file and change its PRIORITY-CLASS value.
3. Alter the ROUTING-CLASS of a request that has already been defined on the Request file and change its ROUTING-CLASS value.

B) Queue the request

Following the identification of the optional QUEUE override parameters, enter this command:

```
QUEUE  #request-name, [(routing-class, priority-class)]
```

BCOM stores this request in the Queue file. There is only one queue for each BCOM environment, so a request that was queued before this one will be processed first.

> ***Note:*** *More than one request-name can be queued in a given QUEUE command. However any override parameters specified will apply to all the requests.*

MVS to NSK Guardian file transfer - Example 1

From a MVS Disk File to a NSK Guardian Disk File, requested from NSK Guardian:

<u>Step I: Define the Request</u>

A) Gathering information

1. **Gather information on NSK Guardian site**
    - The local location is a NSK Guardian labelled TAND1. This name must be configured as location name in the NSK Guardian configuration file. By default, the local-location name is assumed by the BFSINT process, but can be overridden.
    - The local file name is $DATA01.MYSUB.DATA.
    - The priority of the BCOM WRITE I/O process will be 150.

2. **Gather information on MVS site**
    - The remote location is an MVS labelled MVSMTL. This name must be configured as remote location name in the NSK Guardian configuration file.
    - The remote file on MVS already exists. Its dataset name is EXAMP1.DSN, and the disposition is Share.

3. **Gather transfer information**
    - The ROUTING-CLASS under which the transfer characteristics will be selected is specified as 06. This ROUTING-CLASS must be a valid class configured under the remote location participating in the transfer.
    - The priority at which this request will wait in the queue is set at 3.

4. **Gather request-chaining information**

    None

B) Set the request attributes

```
>RUN BFSINT $FSMON

~SET FILE-TYPE       RF
~SET LOCAL-LOCATION  %TAND1
~SET LOCAL-FILE      $DATA01.MYSUB.DATA
~SET LOCAL-ATTR1     PROCESS-PRIORITY = 150
~SET REMOTE-LOCATION %MVSMTL
~SET REMOTE-FILE     EXAMP1.DSN
~SET REMOTE-ATTR1    DISP=SHR
~SET ROUTING-CLASS   6
~SET PRIORITY-CLASS  3
```

C) Add the request

```
~ ADD  #REQUP01
```

<u>Step II: Activate the Request</u>

A) Define the queue overrides

There are no specific overrides in this example.

B) Queue the request

```
~ QUEUE  #REQUP01
```

MVS to NSK Guardian: - Example 2

Receive data transferred from an MVS tape file to a NSK Guardian process:

Step I: Define the Request

A) Gathering information

1. **Gather information on NSK Guardian site**
   - The local location is a NSK Guardian labelled TAND1. This name must be configured as location name in the NSK Guardian and MVS configuration files. In the MVS config file, TAND1 appears as one of the remote locations.
   - The local process name will be $PROG.
   - The BCOM I/O process priority will be 170.
2. **Gather information on MVS site**
   - The remote location is an MVS labelled MVSMTL. This name must be configured as location name in the NSK Guardian and MVS configuration files. In the NSK Guardian config file, MVSMTL appears as one of the remote locations.
   - The remote file on MVS is a tape file.
   - The file has the DCB of a print file.
   - The disposition of this file is old.
   - The remote tape dataset name is "EXAMPLE.NO1."
   - The file will be stored as the first file sequence on the tape.
   - The tape's UNIT name is "TAPE."
3. **Gather transfer information**
   - All fields will be converted from EBCDIC to ASCII.
   - The ROUTING-CLASS under which the transfer characteristics will be selected is specified as 06. This ROUTING-CLASS must be a valid class configured under the remote location participating in the transfer.
   - The priority of this request in the queue is set at 3.
4. **Gather request-chaining information**

   None

B) Set the request attributes

```
READY
>RUN BFSINT $BMON
~SET TYPE            RF
~SET LOCAL-LOCATION  %TAND1
~SET LOCAL-FILE      $PROG
~SET LOCAL-ATTR1     PROCESS-PRIORITY=170
~SET REMOTE-LOCATION %MVSMTL
~SET REMOTE-FILE     EXAMPLE.NO1
~SET REMOTE-ATTR01   DCB=(LRECL=133,BLKSIZE=13300,RECFM=FBA)
~SET REMOTE-ATTR02   DISP=(OLD)
~SET REMOTE-ATTR03   LABEL=(1,SL)
~SET REMOTE-ATTR04   UNIT=TAPE
~SET ROUTING-CLASS   6
~SET PRIORITY-CLASS  3
```

Notes: *After the remote ATTRn designator, syntax is identical to MVS JCL conventions.*

The order of the ATTRn designators is not prescribed. Any order will be accepted.

C) Add the request

```
~ ADD  #REQUP02
```

Step II: Activate the Request

A) Define the queue overrides

None for this example.

B) Queue the request

```
~ QUEUE  #REQUP02
```

Data Conversion Utility

BCOM provides a feature, which gives the user selective control over ASCII/EBCDIC and EBCDIC/ASCII conversion of records targeted at platforms using different character sets. For all types of transfers, BCOM will, by default, convert all fields in a given record image from one character set to the other.

For instance, NSK Guardian implements the ASCII character set, while MVS and other vendors use the EBCDIC character set. When performing a transfer from NSK Guardian to MVS, records will be converted from ASCII to EBCDIC automatically

In order to suppress the default character set conversion, for instance when binary data is being transferred, BCOM provides the SET FIELD command.

Using SET FIELD

To transfer binary data from either the NSK Guardian or from the EBCDIC node as it is, use the SET FIELD option of the request definition command.

The syntax to use when entering this command is as follows:

```
SET FIELD [offset, length [, offset, length...]] | ALL
```

where:

offset	specifies the numeric displacement, or how far into the record character set suppression should begin. An offset of 0 is valid.
length	indicates the number of characters from the offset position, for which character set conversion should be suppressed.
ALL	means no conversion will be done.

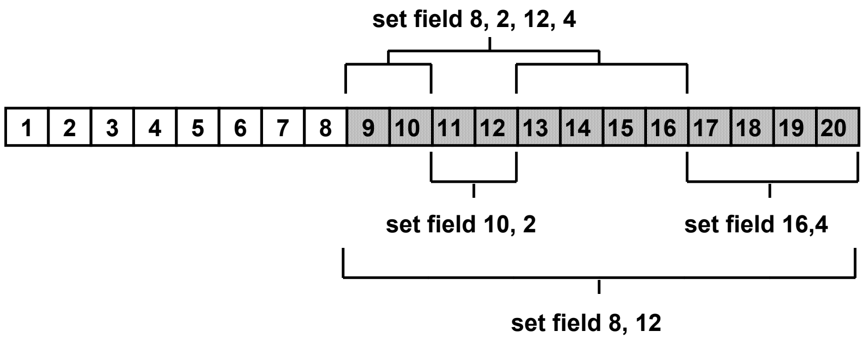
The SET FIELD command can be specified with any file transfer type request. The commands are cumulative, and will reflect this accumulation when the user executes an INFO command against the request. For example,

```
SET FIELD 8, 2, 12, 4
SET FIELD 16, 4
SET FIELD 10, 2
```

is the same as

```
SET FIELD 8, 12
```

as can be seen with the diagram below.



To reactivate conversion for the indicated number of bytes from the indicated offset, use the RESET FIELD command. For example:

```
SET FIELD 8, 12
RESET FIELD 8, 2
```

results in the equivalent of:

```
SET FIELD 10, 10
```

Conversion Suppression: Example

The following example illustrates a conversion, using the SET FIELD command.

```
SET TYPE            SF
SET LOCAL-LOCATION  %TAND1
SET LOCAL-FILE      $DATA01.SUBV.TEST1
SET REMOTE-LOCATION %MVSMTL
SET REMOTE-FILE     MVS.DEST.FILE
SET REMOTE-ATTR01   DISP=SHR
SET ROUTING-CLASS   3
SET PRIORITY-CLASS  5
SET FIELD           8,2,12,4
ADD #RQSTNO1
```

Explanation: In the above example, a request is being made to send a file from NSK Guardian to MVS. Conversion is being suppressed at offset 8, for a length of 2 characters (SET FIELD 8, 2) and at offset 12 for a length of 4 characters (SET FIELD 12, 4).

CheckPoint / ReStart Facilities

The CHECKPOINT/RESTART commands prevent the beginning of a file from being unnecessarily retransmitted in the event of transfer abend. This facility is useful, particularly when large files must be transferred over slower speed lines.

For example, under normal conditions, if a transfer that usually takes four hours to complete abends after three and a half hours, the transfer would have to be restarted from the beginning. The SET CHECKPOINT option of the request definition can be used to prevent this unnecessary retransmission. When specified, it will guarantee that only that portion of the file after the last checkpointed value will be retransmitted.

For instance, the SET CHECKPOINT 50 command instructs BCOM to save the relative record number or key of the last record read or written every 50 records.

When a SET CHECKPOINT [value] command has been executed for a request, once that request has been queued it can then be made eligible for restarting. This is done via the QUEUE/RESTART command.

For example, if request #REQABC was defined with a value of 50 via the SET CHECKPOINT command, and the request ended abnormally, the request could be restarted by entering the following:

```
QUEUERESTART #REQABC
```

This command tells BCOM to queue the request #REQABC for restart from the point where the failure occurred.

Further Considerations

There are some further considerations to take into account when using the CHECKPOINT/RESTART facility, as follows:

- CHECKPOINT/RESTART is not recommended when using BCOM over a SNAXLINK channel-attached device.
- The QUEUE/RESTART command must only be requested for a request that has previously been queued with the SET CHECKPOINT option.
- The SET CHECKPOINT checkpoint-value must be greater than or equal to 50.
- The CHECKPOINT/RESTART facility is not supported for SQL tables or NSK Guardian Unstructured files.
- THE QUEUE/RESTART command should only be attempted if the two files participating in the transfer have been left unchanged as a result of the abnormal end.

- For a file transfer to a remote location where SET CHECKPOINT has been specified, a checkpoint operation will be performed when the checkpoint value has been reached. If, however, the checkpoint value has been reached but the next WINDOW-COUNT has not yet been reached, BCOM will delay the checkpoint operation until it receives acknowledgement from the remote partner, which takes place every WINDOW-COUNT blocks.
- This ensures that the checkpoint taken is reflected on both sides of the transfer operation. For further information, refer to the WINDCNT parameter of BCOM configuration JCL stream in the *BCOM on NSK Guardian Installation & Operations Guide*.
- To restart the transfer from square one - e.g. from the last saved checkpoint - the keyword COLD can be added to the QUEUE RESTART command.
- When RESTARTing the transfer of a key sequenced file which was originally queued with CHECKPOINTing specified, you should be conscious of possible collating sequence differences between the ASCII and EBCDIC character sets for ASCII-based platforms. This is only a problem at RESTART time, due to the file repositioning logic executed prior to restarting the transfer.
- If the QUEUE/RESTART option is used on a key-sequenced file being transferred to the NSK Guardian and the APPEND parameter is specified, the result will be accurate only if the first key restarted from the source file is greater than the last key checkpointed in the target file; otherwise, unpredictable results are to be expected.
- This is due to the file repositioning logic executed at RESTART time. Some records may have been written to the target file after the last successful CHECKPOINT. At QUEUE/RESTART, incoming records are compared against these records "written since last checkpoint" and dropped if equal; otherwise BCOM assumes it can restart writing records. These records initially sent after restarting the download may have key values less than the record on the target platform last compared. This will result in restart errors. The circumstances under which this happens are rare.
- Source files defined as processes for RESTART, should pass all data from the beginning. (BCOM is responsible for the RESTART processing).
- On NSK Guardian, CHECKPOINT/RESTART is supported for user processes defined in the SET LOCAL-FILE or SET REMOTE-FILE request. For transfers to a NSK GUARDIAN location, the BCOM I/O process will start writing records at the last successfully checkpointed record. For requests originating from a NSK Guardian location, the BCOM I/O process will skip any record passed to it by the user's process until the checkpoint value is reached. Only then will BCOM restart transmission, starting from the last successfully checkpointed record.
- On MVS, the user should pay special attention to the DISP setting when the Remote-location is specified as MVS. If the target MVS file already exists, it is recommended that the user specifies SET REMOTE-ATTRnn DISP=SHR. If the target MVS file does not exist, the user should specify SET REMOTE-ATTRnn DISP=(NEW, CATLG, CATLG). This to ensure that the contents of the uncompleted transfer are preserved after abnormal termination of the original checkpointed request.

- The Auto-Restart facility of BCOM has a different technique for handling recoverable errors during data transfer. For more information, please refer to Auto-Restart in the Error Recovery section of this manual.

BCOM Job Submission

This chapter discusses how to execute BCOM Job Submissions. As such, it illustrates the basic categories of Job submission requests supported by the product and provides several examples of how to setup, initiate and control such submissions.

The section is organized according to the following subsections:

1. Defining a Job Submission Request explains the general principles of defining a job submission request, based on where the job will run and where the job control language or commands reside.
2. Procedures for Local Job Submission explain how to execute a TACL job on a local NSK Guardian node, with TACL commands that reside on the NSK Guardian or on a remote node;
3. Procedures for Remote Job Submission explain how to execute a job on a remote node, with control language commands that reside on the NSK Guardian or on a remote node.

Defining a Job Submission Request

A job submission request can be defined as local (LJ) or remote (RJ). The JOB submission's definition (local or remote) indicates where the job will run; while the location of the input command stream for this job is determined by the file keyword (local or remote). If the input job file is remote and the job is local (or vice versa), a file transfer will take place automatically between the two platforms.

In the following example, the LJ identifier indicates that the job will run locally, while the presence of a remote-file parameter makes BCOM look for the job file on a remote platform.

Example:

```
SET TYPE            LJ
SET LOCAL-LOCATION  %TAN1
SET REMOTE-LOCATION %MVS1
SET REMOTE-FILE     MVS-FILE-NAME
SET ...
```

Job Transfer Types

The chart on below illustrates the possible combinations of job submission request parameters and file keywords for the different types of Job Submission requests, as defined FROM THE NSK GUARDIAN PLATFORM; it also indicates when each parameter is to be used.

	REQUEST TYPE	**LJ (Local Job)**		**RJ (Remote Job)**	
	JOB FILE LOCATION	**LOCAL JOB FILE**	**REMOTE JOB FILE**	**LOCAL JOB FILE**	**REMOTE JOB FILE**
	local-location	√ [1]	√ [1]	√ [1]	√ [1]
	remote-location	Not applicable	√	√	√
	local-file	√	**not allowed**	√	**not allowed**
	remote-file	**not allowed**	√	**not allowed**	√
P	**local-attr [3]**	outfile= , job-waited= , process-priority=	outfile= , job-waited= , process-priority=	Not applicable	Not applicable
A	**remote-attr**	Not applicable	√	[4]	[4]
R	**routing-class**	Not applicable	√	√	√
A	**priority-class**	√ [2]	√ [2]	√ [2]	√ [2]
M	**normal-end**	Optional	Optional	Optional	Optional
S	**abnormal-end**	Optional	Optional	Optional	Optional
	auto-restart	Optional	√	√	Optional
	checkpoint	Not applicable	Not applicable	Not applicable	Not applicable
	compression	Not applicable	Optional	Optional	Not applicable
	field	Not applicable	Optional	Optional	Not applicable

√ required parameter

[1] Also defaults to the local location of the BFSINT's monitor

[2] This parameter defaults to a value of '5'

[3] Optional NSK Guardian environment parameters

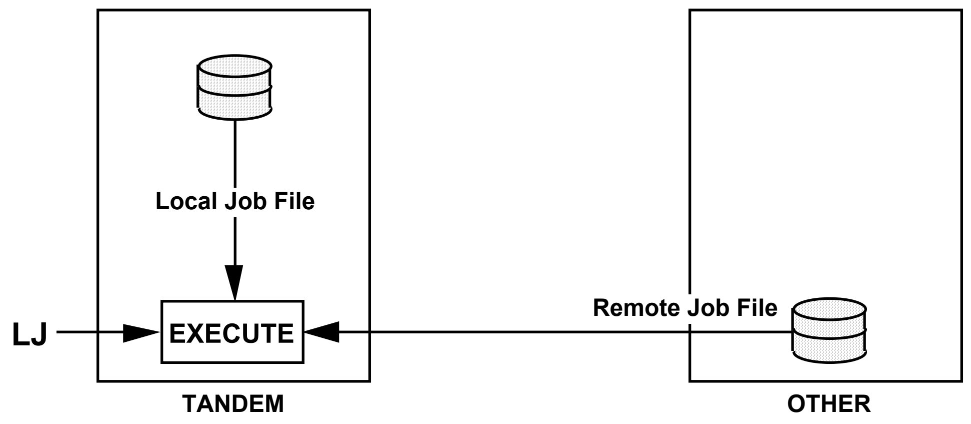
[4] Platform specific parameter

Note: *As you can see, the local-file and remote-file parameters, which indicate the name of the job file, are mutually exclusive.*

Procedures for local job submission

This section describes the BCOM usage conventions and guidelines for submitting a local NSK Guardian job using a job file that resides either on a local or a remote platform.

Local job submission



The JOB executes locally; the JOB submission request is initiated locally; the JOB file data resides either locally, or at this remote location.

NSK Guardian TACL jobs

Whereas, on an MVS machine, the JOB is a collection of JCL organized into a JOB stream, on a NSK Guardian machine, the JOB is a collection of commands suitable for processing by a TACL program. This job stream is usually called an OBEY file or a command file (it can be a TACL macro).

Such job streams are normally stored on disk and are referred to as the JOB file, in the following discussion.

NSK Guardian OBEY files are a particular case of job submission. OBEY files is a generic category that includes TACL command files. These OBEY files can reside on a NSK Guardian node or may be imported from any remote location. They will be processed by a TACL program, as specified in the NSK Guardian configuration file for BCOM (refer to the BCOM on NSK Guardian Installation & Operations Manual).

BCOM actually creates a TACL process and passes the user's file TACL as the IN file and what the user specifies in the BCOM OUTFILE command as the OUT file.

This facility is useful for running NSK Guardian applications together with BCOM transfer requests. Thus you can intermix file transfers with NSK Guardian application processes in order to implement "distributed" applications across environments.

About the Command Interpreter: The NSK Guardian process that will execute a command file is defined at installation time. It is usually a standard TACL process.

> ***Note 1:*** *To select the proper interpreter, please refer to the Job-Submission Module of the <Module-Definition> in the configuration file of BCOM.*

The OUT file to be used by this process needs to be known.

Please note, however, that the file need not exist at request definition time — only when the request is QUEUED for execution.

> ***Note 2:*** *NSK Guardian supports WAITED job submissions. You can submit jobs and then wait for their completion. This allows BCOM to drive the Normal-End or Abnormal-End of a request based on the job's completion STATUS rather than scheduling STATUS.*

Defining a local NSK Guardian TACL job with a local job file

This section describes the BCOM usage conventions and guidelines for submitting a local NSK Guardian job using a job file that resides on the local platform.

Step I: Define The Request

A. Information gathering

Locate the JOB file: By definition, the JOB file resides locally. The LOCAL-FILE keyword is used to identify the JOB input data, and the corresponding LOCATION keyword indicates the name of the local node, as BCOM recognizes it. The LJ request parameter indicates that the job will run on the local node.

For a disk file, the fully qualified disk file name is required. For a tape file, the Tape unit must be known. For a DNS ALIAS on NSK Guardian, the name must be present in the distributed database of names. If a named process is to act as file name, then its process ID name must be known.

Decide on the I/O process priority: BCOM assigns a default priority to the I/O module (e.g. TACL) that is responsible for reading and processing JOB file data on a NSK Guardian platform. You may modify the priority of that process through the PROCESS-PRIORITY keyword. Note that this environment variable is different and independent from the priority-class parameter of the BCOM software (see table on page 3).

Select a Job Report Log: If the job is a FUP or some other program that generates an output listing, then add an OUTFILE parameter to direct this output to a spooler location of your choice.

B. Set the Request Attributes

Using the information you obtained through A, and according to the rules explained in the first pages of this chapter, set the required parameters as follows:

```
SET TYPE           LJ
SET LOCAL-LOCATION %local-platform-identifier
SET LOCAL-FILE     NSK Guardian-file-name
```

or

```
SET LOCAL-FILE     DNS-ALIAS=alias-name
SET LOCAL-ATTR1    OUTFILE=spooler-location
SET PRIORITY-CLASS n
```

C. Add the Request to the Request File

Immediately following the SET commands (within the same BFSINT session), enter the ADD command with the request name:

```
ADD  #request-name
```

Step II: Activate The Request

The request may be activated at any time, within the same BFSINT session or at a later time. The JOB file must be present on NSK Guardian. If it comes from a process, this process must be started.

A. Identify the queue overrides

None applicable.

B. Queue the request

```
QUEUE #request-name
```

BCOM stores this request in the Queue file. There is only one queue for each BCOM environment, so any request that was queued before on the local platform and do not require communication processes will be processed first, except if some of these have a lower priority class than the one entering.

When your JOB has run, BCOM automatically queues the request whose name was specified with the SET NORMAL-END or SET ABNORMAL-END parameter (if any).

Defining a local NSK Guardian TACL job with a remote job file

This section describes the BCOM usage conventions and guidelines for submitting a local NSK Guardian job using a job file that resides on a remote platform.

Step I: DEFINE THE REQUEST

A. Information gathering

Locate the JOB file: By definition, the JOB file resides on a remote platform. The REMOTE-FILE keyword is used to identify the JOB input data, and the corresponding LOCATION keyword indicates the name of the local node, as BCOM recognizes it. You have also to be aware of the rules governing the file names on the remote platform. The LJ request parameter indicates that the job will run on the local node.

B. Set the Request Attributes

Using the information you obtained through A, and according to the rules explained in the first pages of this chapter, set the required parameters as follows:

```
SET TYPE            RJ
SET LOCAL-LOCATION  %local-platform-identifier
SET REMOTE-LOCATION %remote-platform-identifier
SET REMOTE-FILE     remote-file-name
SET REMOTE-ATTR1    remote-attr            (ex.: DISP=SHR)
SET LOCAL-ATTR1     outfile=spooler-location
SET ROUTING-CLASS   routing-class
SET PRIORITY-CLASS  n
SET AUTO-RESTART    yes | no
```

C. Add the Request to the Request File

Immediately following the SET commands (within the same BFSINT session), enter the ADD command with the request name:

```
ADD #request-name
```

Step II: Activate the request

The request may be activated at any time, within the same BFSINT session or at a later time. The JOB file must be present on NSK Guardian. If it comes from a process, this process must be started.

A. Identify the queue overrides

None applicable.

B. Queue the request

```
QUEUE #request-name
```

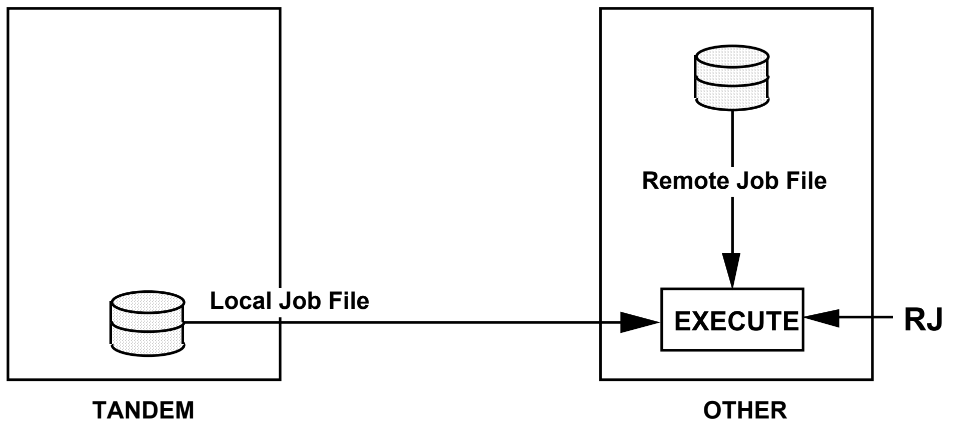
BCOM stores this request in the Queue file. There is only one queue for each BCOM environment, so any request that was queued before on the local platform will be processed first, except if some requests are still waiting for the same communication processes and have a lower priority class than the one entering, or if the entering request asks for available communication processes that are different than those the other requests are waiting for.

When your JOB has run, BCOM automatically queues the request whose name was specified with the SET NORMAL-END or SET ABNORMAL-END parameter (if any).

Procedures for remote job submission

This section describes the BCOM usage conventions and guidelines for submitting a remote job using a job file that resides either on a local or a remote platform.

Remote job submission



The JOB executes at a remote location; the JOB submission request is initiated at this same remote location; the JOB file data resides either locally, or at this remote location.

Defining a remote job with a local job file

This section describes the BCOM usage conventions and guidelines for submitting a remote job using a job file that resides on the local platform.

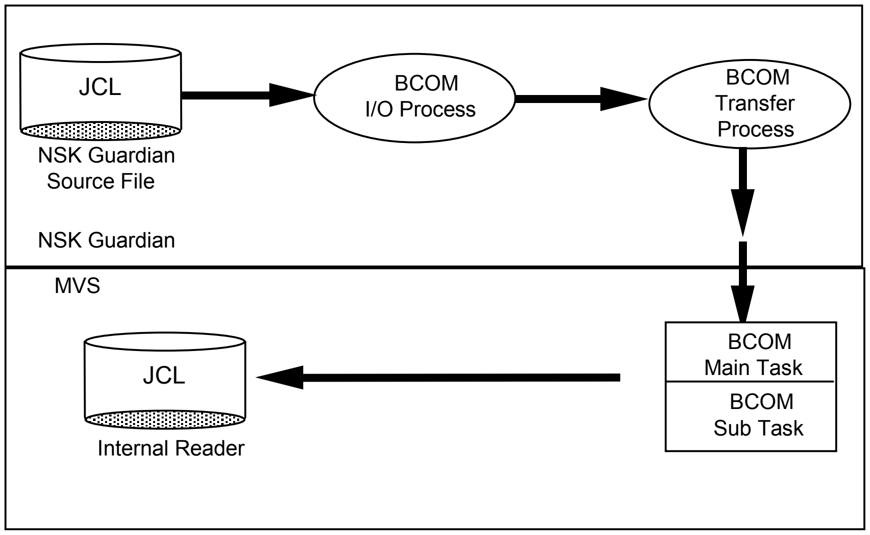
Step I: Define the request

A. Information gathering

Locate the Job File: By definition, the JOB file resides locally. The LOCAL-FILE keyword is used to identify the JOB input data, and the corresponding LOCATION keyword indicates the name of the local node, as BCOM recognizes it. The RJ request parameter indicates that the job will run on a remote node.

There is no specific information to be known if the remote platform is an MVS, since a JOB file executing on an MVS will be submitted to the platform's job scheduler (in this case, the MVS internal reader).

Job submission from NSK Guardian to MVS



B. Set the Request Attributes

Using the information you obtained through A, and according to the rules explained in the first pages of this chapter, set the required parameters as follows:

```
SET TYPE            RJ
SET LOCAL-LOCATION  %local-platform-identifier
SET REMOTE-LOCATION %remote-platform-identifier
SET LOCAL-FILE      local-file-name
SET ROUTING-CLASS   routing-class
SET PRIORITY-CLASS  n
SET AUTO-RESTART    yes | no
```

C. Add the Request to the Request File

Immediately following the SET commands (within the same BFSINT session), enter the ADD command with the request name:

```
ADD #request-name
```

Step II: Activate the request

The request may be activated at any time, within the same BFSINT session or at a later time. The JOB file must be present on NSK Guardian. If it comes from a process, this process must be started.

A. Identify the queue overrides

None applicable.

B. Queue the request

```
QUEUE #request-name
```

BCOM stores this request in the Queue file. There is only one queue for each BCOM environment, so any request that was queued before on the remote platform will be processed first, except if some requests are still waiting for the same communication processes and have a lower priority class than the one entering, or if the entering request asks for available communication processes that are different than those the other requests are waiting for.

When your JOB has run, BCOM automatically queues the request whose name was specified with the SET NORMAL-END or SET ABNORMAL-END parameter (if any).

Defining a remote job with a remote job file

This section describes the BCOM usage conventions and guidelines for submitting a remote job using a job file that resides on a remote platform.

Step I: Define the request

A. Information gathering

Locate the Job File By definition, the JOB file resides on a remote node. The REMOTE-FILE keyword is used to identify the JOB input data, and the corresponding LOCATION keyword indicates the name of the remote node, as BCOM recognizes it. The RJ request parameter indicates that the job will run on the remote node.

B. Set the Request Attributes

Using the information you obtained through A, and according to the rules explained in the first pages of this chapter, set the required parameters as follows:

```
SET TYPE            RJ
SET LOCAL-LOCATION  %local-platform-identifier
SET REMOTE-LOCATION %remote-platform-identifier
SET REMOTE-FILE     remote-file-name
SET REMOTE-ATTR1    remote-attr  (ex.: DISP=SHR)
SET ROUTING-CLASS   routing-class
SET PRIORITY-CLASS  n
```

C. Add the Request to the Request File

Immediately following the SET commands (within the same BFSINT session), enter the ADD command with the request name:

```
ADD #request-name
```

Step II: Activate the request

The request may be activated at any time, within the same BFSINT session or at a later time. The JOB file must be present on NSK Guardian. If it comes from a process, this process must be started.

A. Identify the queue overrides

None applicable.

B. Queue the request

```
QUEUE #request-name
```

BCOM stores this request in the Queue file. There is only one queue for each BCOM environment, so any request that was queued before on the remote platform will be processed first, except if some requests are still waiting for the same communication processes and have a lower priority class than the one entering, or if the entering request asks for available communication processes that are different than those the other requests are waiting for.

When your JOB has run, BCOM automatically queues the request whose name was specified with the SET NORMAL-END or SET ABNORMAL-END parameter (if any).

Backup/Restore

This chapter discusses how to execute BCOM BACKUP/RESTORE. As such, it illustrates the basic categories of backup/restore requests supported by the product and provides several examples of how to setup, initiate and control such requests.

The section is organized according to the following subsections:

1. Online Backup/Restore explains how to execute backup and restore over the network.
2. Examples of Backup/Restore Request shows some request examples.
3. Considerations for Backup/Restore contains general guidelines and names the supported Backup/Restore options.

On-line backup/restore

This section describes the BCOM usage conventions and guidelines for submitting BACKUP-RESTORE from the local location to a particular remote location.

The next section, "*Stand-alone restore*", presents a way to restore when the network is not available.

BCOM uses NSK Guardian's Backup/Restore utility to do the actual backup/restore. BCOM either receives the data or sends the data to these utilities just as if it were the tape device. The data from a backup request is in a format internal to NSK Guardian's backup utility and is therefore not usable at the remote site.

Backup request

A backup request is defined once, and stored in the BCOM request file by an ADD command. Once defined, the request is executed by the QUEUE command.

There are two types of request commanding a Backup of NSK Guardian disk files:

- FB (File Backup) requests are defined and initiated from BCOM, on NSK Guardian side
- RB (Remote Backup) requests are defined and initiated from BCOM, on MVS side

A backup request contains two main groups of information:

- The NSK Guardian information is essentially the command to be executed by the NSK Guardian Backup utility, it may also specify the output and the process priority for the Backup process. For a type FB request, this information is entered in the Local Attributes of the request (in the Remote Attributes of a type RB request).
- The MVS information describes the MVS file which will contain the backup object, equivalent of a "tape backup". For a type FB request, this information is entered in the Remote File and Remote Attributes of the request.

NSK Guardian information

The Backup command is specified through the BRPARAMS keyword of the request Attributes, in two alternative ways, depending on the size of the Backup command:

1. The Backup command itself is entered in one to ten BRPARAMS lines. The syntax is the standard Backup syntax, with two exceptions:
   - The tape device is omitted, the command starts by the file-set to backup
   - Some Backup options are not supported (see later in this section)
2. The Backup command is included in a user file, and the file name is specified in a unique BRPARAMS line: SET LOCAL-ATTR01 BRPARAMS=INFILE <file-name>.
   - When the INFILE facility is chosen, the user has to prepare an edit file containing a complete Backup command with the DEFINE =BCOM-TAPE as tape device - even for Guardian environments without tape label support.
   - Some Backup options are not supported (see later in this section)

MVS information

About the destination file:

Default DATA CONTROL BLOCK parameters will be set by BCOM as follows: (LRECL=4078, BLKSIZE=4082, RECFM=VB). You can change the default block size, up to 32000, by specifying the preferred DCB in a remote attribute. For example:

```
SET REMOTE-ATTR06 DCB=(LRECL=4078, BLKSIZE=20000, RECFM=VB)
```

About the destination tape(s):

Please note that with BCOM, all NSK Guardian data being backed up to MVS will be stored into a single logical dataset. This dataset can span several physical tape volumes: by allocating to a tape unit and requesting the creation of a new dataset (NEW or SCRATCH), the MVS operating system will find and allocate as many - up to the prescribed maximum, that is - physical tape volumes as required in order to contain the NSK Guardian data.

Please refer to the VOL attribute in the SET command of the request definition for how to change this maximum. The default maximum is 5 volumes; to use more than five tape volumes, add the following attribute to your backup request definition: , where mm is the new maximum number of tape volume(s) associated with this dataset.

Adding a disposition of CATLG to your request definition will cause the backup dataset to be recorded into the MVS catalog. This will greatly facilitate the restore operation: specifying only the dataset name and a disposition of OLD in your request definition will be enough for MVS to automatically locate and mount all the tape volume(s) associated with this dataset.

Restore request

There are two types of request commanding a Restore of NSK Guardian disk files:

- FR (File Restore) requests are defined and initiated from BCOM, on NSK Guardian side
- RR (Remote Restore) requests are defined and initiated from BCOM, on MVS side

As for backup the NSK Guardian information is essentially the Restore command to be executed by the NSK Guardian Restore utility, and the MVS information names the MVS file containing the backup object to read.

The Restore command is entered through the same BRPARAMS keyword of the BCOM request attribute(s).

Note that INFILE is also available for restore requests.

STARTFILE *Guardian-file-name* can be coded in the remote attributes of an FR request, to direct the MVS to skip the transfer of NSK Guardian files before the specified filename.

Examples of backup/restore requests

Example 1: Generic backup

Using the information from the precedent page, you can now enter the backup request definition using the BCOM INTERFACE (BFSINT) program. The following SET COMMANDS are used to define the request's attributes.

Set the request attributes

Request Type:

```
~ SET TYPE FILE-BACKUP
```

NSK Guardian Information: (may span across many local attributes)

```
~ SET LOCAL-LOCATION  <%local location name>
~ SET LOCAL-ATTR01 BRPARAMS=<file-list, backup options>
~ SET LOCAL-ATTR02 BRPARAMS=<file-list, backup options>
```

MVS Information: (REMOTE- ATTRIBUTES FOR REMOTE FILE)

```
~ SET REMOTE-LOCATION <%remote location name>
~ SET REMOTE-FILE     <MVS dataset name>
~ SET REMOTE-ATTR01 DCB=<MVS file DCB info>
~ SET REMOTE-ATTR02 DISP=<MVS dataset disposition>
~ SET REMOTE-ATTR03 VOL=<MVS volumes>
~ SET REMOTE-ATTR04 LABEL=<MVS tape label info>
~ SET REMOTE-ATTR05 SPACE=<MVS space info>
~ SET REMOTE-ATTR06 UNIT=<MVS file unit info>
```

General Information:

```
~ SET ROUTING CLASS <99> (as defined in configuration)
~ SET FIELD ALL          (as no conversion is recommended)
~ SET COMPRESSION   YES  (compression is suggested)
~ SET NORMAL-END    <request-name>
~ SET ABNORMAL-END  <request-name>
```

Add the request

Immediately following the SET commands (within the same BFSINT session) you enter the ADD command with the request name

```
~ ADD <request name>
```

When this command is issued, the request is added with all it attributes that were previously set.

Activate the request

The request may be activated at any time, within the same BFSINT session or at a later time.

```
QUEUE <request-name>
```

Example 2: Backup an entire NSK Guardian subvolume to MVS

I. Define the Request

A) Get ready

- Backup all the files on subvolume $DATA01.MYSUB
- The destination file on MVS already exists.
- Its dataset name is EXAMP1.DSN
- Include all the audited disk files
- As an option, if this transfer completes normally, #BACK02 will be automatically scheduled.

B) Set the request attributes

```
>RUN BFSINT
~ SET TYPE             FILE-BACKUP
~ SET LOCAL-LOCATION   %TAND1
~ SET LOCAL-ATTR01     OUTFILE=$S.#BACKUP
~ SET LOCAL-ATTR02     BRPARAMS=($DATA01.MYSUB.*),AUDITED ,
~ SET LOCAL-ATTR03     BRPARAMS=LISTALL
~ SET REMOTE-LOCATION %MVS1
~ SET REMOTE-FILE      EXAMP1.DSN
~ SET REMOTE-ATTR01    DISP=SHR
~ SET ROUTING-CLASS    1
~ SET FIELD            ALL
~ SET COMPRESSION      YES
~ SET NORMAL-END       #BACK02
```

C) Add the request

```
~ ADD #BACK01
```

II. Activate the Request

Queue the request

```
~ QUEUE #BACK01
```

Example 3: Use a separate INFILE for backup

I. Define the Request

A) Get ready

Create the INFILE $DATA.BACKUP.INFILE containing:

```
=BCOM-TAPE, ($DATA01.MYSUB.*),AUDITED, LISTALL
```

B) Set the request attributes

```
>RUN BFSINT
~ SET TYPE            FILE-BACKUP
~ SET LOCAL-LOCATION  %TAND1
~ SET LOCAL-ATTR01    OUTFILE=$S.#BACK02
~ SET LOCAL-ATTR02    BRPARAMS=INFILE $DATA.BACKUP.INFILE
~ SET REMOTE-LOCATION %MVS1
~ SET REMOTE-FILE     EXAMP2.DSN
~ SET REMOTE-ATTR01   DISP=(NEW,CATLG,DELETE)
~ SET REMOTE-ATTR02   UNIT=TAPE
~ SET REMOTE-ATTR03   DSN=TANDEM.BACKUP.BACK02(+1)
~ SET REMOTE-ATTR04   DCB=(RECFM=VB,LRECL=4078,BLKSIZE=32000)
~ SET ROUTING-CLASS   1
~ SET FIELD           ALL
~ SET COMPRESSION     YES
~ SET NORMAL-END      #BACK02
```

C) Add the request

```
~ ADD #BACK01
```

II. Activate the Request

Queue the request

```
~ QUEUE #BACK01
```

Example 4: Generic restore

Using the information from Step a, you can now enter the restore request definition using the BCOM INTERFACE (BFSINT) program. The following SET commands are used to define the request's attributes.

I. Define the Request

Set the request attributes

Request Type:

```
~ SET TYPE FILE-RESTORE
```

NSK Guardian Information : (may span across many local attributes)

```
~ SET LOCAL-LOCATION <%local location name>,
~ SET LOCAL-ATTR01 BRPARAMS=<file-list, restore options>,
~ SET LOCAL-ATTR02 BRPARAMS=<file-list, restore options>
```

MVS Information : (remote attributes used to define remote file)

```
~ SET REMOTE-LOCATION <%remote location name>
~ SET REMOTE-FILE     <MVS dataset name>
~ SET REMOTE-ATTR01 DCB=<MVS DCB info>
~ SET REMOTE-ATTR02 DISP=<MVS dataset disposition>
~ SET REMOTE-ATTR03 VOL=<MVS volumes>
~ SET REMOTE-ATTR04 LABEL=<MVS tape label info>
~ SET REMOTE-ATTR05 SPACE=<MVS space info>
~ SET REMOTE-ATTR06 UNIT=<MVS file unit info>
```

General Information:

```
~ SET ROUTING-CLASS <99>  (as defined in configuration)
~ SET FIELD ALL           (as no conversion is recommended)
~ SET COMPRESSION   YES  (compression is suggested)
~ SET NORMAL-END    <request-name>
~ SET ABNORMAL-END  <request-name>
```

Add the request

Immediately following the SET commands (within the same BFSINT session) you enter the ADD command with the request name.

```
~ ADD <request name>
```

When this command is issued, the request is added with all it attributes that were previously set.

II. Activate the request

The request may be activated at any time, within the same BFSINT session or at a later time.

Queue the request

```
~ QUEUE <request-name>
```

Example 5: Restore one file from the previous backup example

I. Define the Request

A) Get ready

- Restore the data file $DATA01.MYSUB .FILE1
- The source file on MVS already exists.
- Its dataset name is EXAMP1.DSN.
- The disposition is SHR.
- Include all the audited disk files and do a listall

B) Set the request attributes

```
>RUN BFSINT
~ SET TYPE            FILE-RESTORE
~ SET LOCAL-LOCATION  %TAND1
~ SET LOCAL-ATTR1     BRPARAMS=($DATA01.MYSUB.FILE1),AUDITED ,
~ SET LOCAL-ATTR2     BRPARAMS=LISTALL
~ SET REMOTE-LOCATION %MVS1
~ SET REMOTE-FILE     EXAMP1.DSN
~ SET REMOTE-ATTR01   DISP=SHR
~ SET ROUTING-CLASS   1
~ SET FIELD           ALL
~ SET COMPRESSION     YES
```

C) Add the request

```
~ ADD #REST01
```

II. Activate the Request

Queue the request

```
~ QUEUE #REST01
```

NSK Guardian's BACKUP and RESTORE options may be specified in the BCOM request definition. Valid options are listed under Backup/Restore considerations on the next page.

Backup/restore considerations

Character Code Translation

In the case of backup data, only the local restore program and the BACKUP/RESTORE Stand-Alone Restore utility will ever read it. No MVS application will be able to process this data as it is stored in the local backup program's internal format. Allowing automatic translation in this case will only waste CPU cycles. For this reason, it is recommended that you turn off automatic translation for all backup and restore requests (SET FIELD ALL).

Notes: *Data translation is not required for backup/restore (i.e. you can set FIELD ALL in the request definition). However, be sure that whatever value has been specified for translation, in a backup request definition, is the SAME specified in the corresponding restore request.*

Beginning with Version C01, there is no need to specify SET FIELD ALL in restore requests, even if the backup object was converted to EBCDIC at backup time.

BLOCKSIZE for NSK Guardian utility

BCOM generates BLOCKSIZE 28 in the backup command passed to the BACKUP or RESTORE program, in order to minimize the number of inter-process messages between BCOM emulating a tape drive and BACKUP or RESTORE program. The user can force a lower value but not a higher.

Compression

Compression usually reduces transfer time and is therefore recommended. Verify that compression is authorized in BCOM configuration, to ensure that the request attribute "SET COMPRESSION YES" will be effective. It is also possible to enforce the compression on all requests, by BCOM configuration.

FastRestore

The FastRestore processing minimizes the transmission of useless data for a partial restore. This functionality is based on the remote attribute STARTFILE=*Guardian-file* of the restore request.

When a restore request executes, the data transfer:

- begins at the beginning of the MVS file to get the backup headers,
- if STARTFILE=*Guardian-file* is present, all files are skipped by the MVS side until the named Guardian-file is reached
- then all files present in the backup are transmitted until the RESTORE program ends reading form the “tape drive” emulated by BCOM.

The STARTFILE attribute is available for normal usage (the value is set by the user) when the BCOM on MVS is version V5.0 or later.

When the BCOM on Tandem V5.1 is configured with the option BACKUP-CATALOG YES, and if the backup catalog was active at backup time, and is active for the restore request, the STARTFILE=*Guardian-file* attribute is generated automatically, based on the file-list to restore and the catalog content.

IO process completion

Each execution of backup or restore request starts IO processes, especially BACKUP or RESTORE. BACKUP or RESTORE programs issue frequently Error and Warning messages during an execution, and signal the presence of such messages by setting a non-zero completion code when they end. Usually the code 1 or 8 is returned for Warning and the code 2 for Errors.

To check easily if a backup was executed without any warning, the local attribute NORMAL-IO-COMPLETION-CODES should accept the only value 0. This attribute is most useful for backup/restore requests, and is documented in the chapter “Follow-on scheduling”.

Checkpoint/Restart

A user can invoke checkpoint/restart capability during backup processing by specifying the CHECKPOINT=nn as a backup (FB) request attribute. This is an option available only when BCOM is configured with the option BACKUP-CATALOG YES.

Checkpoint/restart is not available for volume mode backups because file information is not provided in that mode and there's no START facility. Checkpoint/restart is not available for the TMF option because there's no START facility in TM/MP.

There is no automatic Checkpoint/restart for Restore requests. But a user can manually modify a Restore request partially executed, by specifying the remote attribute STARTFILE = filename.

The command "SET CHECKPOINT number-of-megabytes" enables the checkpoint/restart and specifies the frequency of checkpoints, in megabytes unit, with a minimum of 250 megabytes. The checkpoint/restart and backup catalog have been designed for a checkpoint interval corresponding to about the capacity of a tape media.

A checkpoint is taken only when a Guardian file is completely stored in the remote backup, consequently very large files will lower the checkpoint frequency.

The general usage of checkpoint restart procedures is presented in the BCOM error recovery chapter. The following specific considerations apply to backup requests.

- A BCOM checkpoint forces a commit to the local backup catalog (CATBK/CATBKFIL files)
- At restart time after an aborted backup, a parameter is added to the backup command passed to the BACKUP program: START *Guardian-file*, where Guardian-file is the file following the last file part of a checkpoint.

The START parameter is added to avoid having to transfer the same data again, but also to avoid reading the same data again at restart time.

This Guardian file might be deleted after the checkpoint and before the restart, by other running applications. If BACKUP does not find the file, it will abort and the request will abort again. If this special condition happens, restarting again (QRESTART) the request will execute BACKUP without the START *Guardian-file* parameter, the rule coded in BCOM is that the START parameter will be tried only once per checkpoint.

Version compatibility

- Backhome and BCOM programs are identical, i.e. backup tapes produced by BackHome and BCOM have exactly the same format.
- Backup tapes on BCOM MVS produced by BackHome (or BCOM) B01, B02, C01,C02 and V4.0 have exactly the same format They can be restored by BCOM V5.1, online or through the standalone restore utility BFSTAPE.
- Backup tapes on BCOM MVS produced by V5.1 can be restored by a previous BCOM version only if no restart procedure was executed during the backup. The stand-alone restore BFSTAPE V5.1 is available for all backups produced by BCOM V5.1.

Backup Options

This version of BFSBKUP process will support File Mode Backup with the following options (See GUARDIAN 90 Operating System Utilities for a full description of the BACKUP options):

`ALTFILE`	
`AUDITED`	
`BLOCKSIZE`	
`IGNORE`	`always be the default`
`INDEXES`	
`LISTALL`	
`NOMYID`	
`NOSAFEGUARD`	
`NOT`	
`OPEN`	
`PAGELENGTH`	
`PART`	
`PARTIAL`	
`PARTONLY`	
`SHAREOPEN`	
`SQLCATALOGS`	*
`START`	(must not be used in a request defined with checkpoint/restart)
`VOL`	
`VOLUMEMODE`	
`WHOLEDISC`	

Restore Options

This version of BFSREST process will support File Mode Restore with the following options (See GUARDIAN 90 Operating System Utilities for a full description of the RESTORE options):

`ALTFILE`	
`AUDITED`	
`AUTOCREATECATALOG`	
`CATALOGS`	
`DETAIL`	
`LISTONLY`	
`INDEXES`	
`KEEP`	
`LISTALL`	
`MAP NAMES`	
`MYID`	
`NOSAFEGUARD`	
`NOT`	
`OPEN`	
`PAGELENGTH`	
`PART`	
`PARTOF`	
`PARTONLY`	
`PHYSVOL`	
`REGISTERONLY`	
`RENAME`	
`SQLCATALOGS`	*
`SQLCOMPILE`	
`START`	
`TAPEDATE`	
`TARGET`	
`TURNOFFAUDIT`	
`VERIFY`	
`VERIFYTAPE`	
`VOL`	
`VOLUMEMODE`	

* Depending on the execution context, the transfer may be aborted. This happens when the Tandem program (BACKUP or RESTORE) issues a backspace operation.

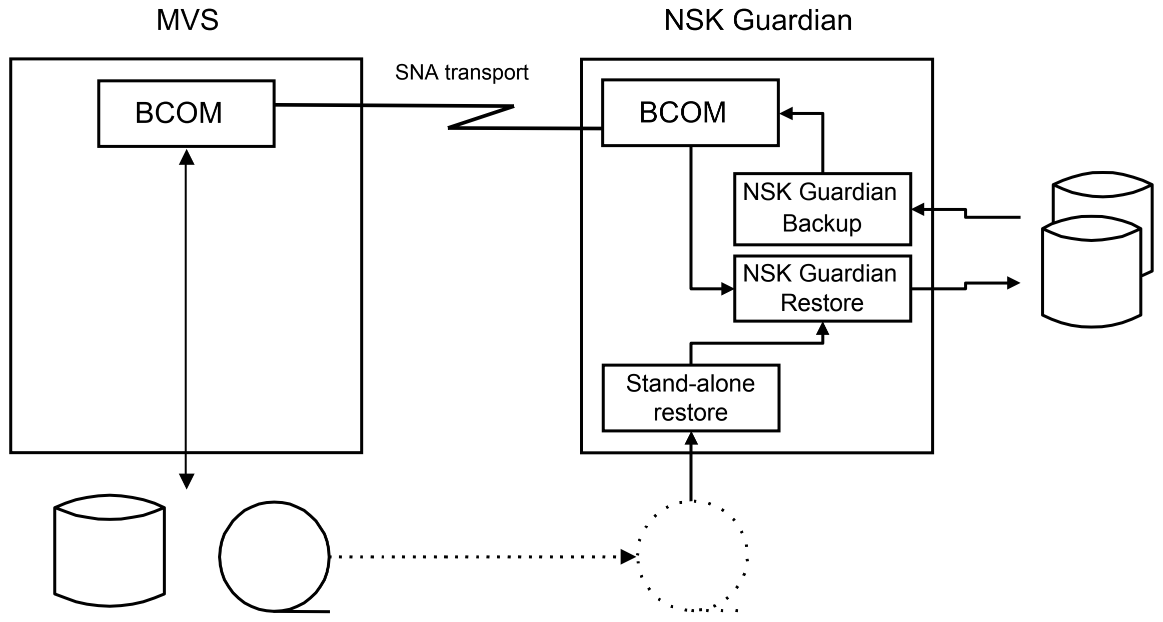
Backup Catalog

Backup Catalog is available only for BackHome/TSM, please refer to the *BackHome/TSM - Installation and User Guide* for more details.

Disaster restore recovery utility

The Disaster Recovery Restore utility can be used to restore a BCOM Backup when the NSK Guardian-MVS connection is not available. This utility is a stand-alone module, which does not require the standard BCOM environment.

Stand-Alone Restore Utility BFSTAPE



In a normal situation, the NSK Guardian BCOM (right) retrieves the backed up data from the MVS repository site, through MVS BCOM (left), using a communication link. If the link is broken, BCOM on NSK Guardian can read the MVS reel tape on its own tape drive, using the Stand-Alone Restore Utility, which passes the tape content to the NSK Guardian stand-alone restore program.

How to run the Disaster Recovery Utility:

The MVS tape containing the NSK Guardian backup stored by BCOM MVS is loaded directly on a NSK Guardian tape drive. The BFSTAPE utility will start the NSK Guardian Restore program, and will emulate a tape drive for the Restore program to give back the regular format of NSK Guardian backup objects.

In a Guardian environment with tape label support, both MVS labelled and non-labelled MVS tapes are supported.

In a Guardian environment without tape label support, only MVS labelled tapes are supported. The tape must contain only a backup object written by BCOM MVS. Multi-volumes files are supported.

BFSTAPE is an independent program run outside the BCOM environment. It should be run via an OBEY file like the STRTREST provided in the BCOM Distribution Tape.

The Product Distribution Tape contains an OBEY file to run the utility: STRTREST. To run BFSTAPE:

- a DEFINE describing the backup object must be created and given to the IN parameter
- the command to be executed by Restore is given in the PARAM RESTORE-STARTUP-COMMAND
- the ASCII-EBCDIC conversion choice set at backup time must be entered in the PARAM CONVERSION
- if the MVS file was created with buffers larger than 32000, that buffersize must be set in the PARAM BFS-BLOCKLEN

DEFINE for the input tape(s)

For Guardian environment supporting tape labels:

- the input DEFINE may have the class TAPE or TAPECATALOG
- if the backup object was written on labelled tapes, LABELS MVS and EBCDIC OFF must be specified

For Guardian environment without tape label support:

- a DEFINE class TAPE must be created, BFSTAPE will use the following DEFINE attributes:

`DEVICE xxxxx`	to know which unit to read
`FILEID xxxx`	to verify the MVS file name (DSN)

Notes: ***VOLUME*** *(volume-list) is not used, but the volume sequence number is verified*

FILEID *is limited to 17 characters. When the MVS DSN is truncated, it is truncated at left: the last 17 characters are stored on the tape.*

PARAM "RESTORE-STARTUP-COMMAND"

This parameter should contain a valid RESTORE command, without the leading device name. The options supported are the same as for online restore; see earlier section in this chapter.

Example:

```
PARAM RESTORE-STARTUP-COMMAND  "$*.*.*,MYID, MAP NAMES $*.*.* TO $DATA*.*.* "
```

PARAM "CONVERSION"

The value must be YES or NO. There is no default.

Normally, backup requests are defined for non-conversion ASCII-EBSDIC (SET FIELD=ALL), and CONVERSION must be NO.

If FIELD attribute was not set in the backup request, CONVERSION must be YES.

A mistake will make the Restore utility reject the "unrecognized tape".

PARAM "BFS-BLOCKLEN"

The value range is 4078-32767. This parameter is optional; the default value is 32000.

This parameter is useful only for backup created with an MVS blocksize larger than 32000 - which is not recommended to be able to execute stand-alone restore.

When the MVS tape was created with blocksize larger than 32000, remote Tandem tape drives (via Guardian Expand) are not supported. BFSTAPE must read the tape on a local tape drive.

Examples of using stand-alone restore

For MVS labelled tapes:

```
DELETE DEFINE =REST
SET    DEFINE CLASS TAPE
SET    DEFINE DEVICE $TAPE
SET    DEFINE LABELS IBM
SET    DEFINE MOUNTMSG "MOUNT TAPE - BCOM RESTORE UTILITY"
SET    DEFINE USE IN
SET    DEFINE VOLUME M22457
SET    DEFINE FILEID ETI01G.TAPE1600
SET    DEFINE EBCDIC OFF
ADD    DEFINE =REST

ADD DEFINE =BCOM-TDM-RESTORE, CLASS MAP, FILE $SYSTEM.SYSTEM.RESTORE
PARAM CONVERSION NO
PARAM RESTORE-STARTUP-COMMAND "(*.*.*.),LISTALL,VOL $DATA.ZZTEST"

RUN BFSTAPE/IN=REST, OUT $S.#OUT,NAME,NOWAIT/
CLEAR ALL
```

For MVS non-labelled tapes:

```
DELETE DEFINE =REST
SET    DEFINE CLASS TAPE
SET    DEFINE DEVICE $TAPE
SET    DEFINE LABELS OMITTED
ADD    DEFINE =REST

ADD DEFINE =BCOM-TDM-RESTORE, CLASS MAP, FILE $SYSTEM.SYSTEM.RESTORE
PARAM CONVERSION NO
PARAM RESTORE-STARTUP-COMMAND "(*.*.*.),LISTALL,VOL $DATA.ZZTEST"

RUN BFSTAPE/IN=REST, OUT $S.#OUT,NAME,NOWAIT/
CLEAR ALL
```

For MVS labelled tapes in a Guardian without support for labelled tapes:

```
DELETE DEFINE =REST
SET    DEFINE CLASS TAPE
SET    DEFINE DEVICE $TAPE
SET    DEFINE LABELS IBM
SET    DEFINE FILEID ETI01G.TAPE1600
ADD    DEFINE =REST

ADD DEFINE =BCOM-TDM-RESTORE, CLASS MAP, FILE $SYSTEM.SYSTEM.RESTORE
PARAM CONVERSION NO
PARAM RESTORE-STARTUP-COMMAND "(*.*.*.),LISTALL,VOL $DATA.ZZTEST"

RUN BFSTAPE/IN=REST, OUT $S.#OUT,NAME,NOWAIT/
CLEAR ALL
```

Remote queuing facility (RQF)

BCOM is designed to facilitate the exchange of information between heterogeneous platforms by using cooperative processing techniques and a peer-to-peer architecture. This design takes into consideration the possibility that one of the platforms running BCOM may be used for centralized management or control - especially in the context of distributed networks or when there is a requirement for lights-out processing.

To help manage distributed platforms, BCOM provides a method for initiating requests that are already defined at a remote location. This method, called the Remote Queuing Facility (RQF), does not rely on indirect mechanisms such as Job Submission or remote logons: it introduces a new type of request definition - the Remote Queue (type=RQ) - which allows for the propagation of queue commands to remote nodes.

The Remote Queuing Facility can be used to broadcast control throughout a network by enabling intermediate node queuing; this can be used advantageously to automate the distribution of software or for the dissemination of corporate information to all of your platforms.

The following subsection discusses how to prepare for and use the Remote Queuing facility of BCOM.

Local and remote queuing

All BCOM transfer activities, except print distribution, must be predefined and stored as request definitions into a repository file called the request file. To initiate them, two forms of request queuing operations can be used:

- the QUEUE and QWAIT commands explicitly schedule a request for execution;
- other mechanisms, such as the NORMAL-END or ABNORMAL-END keywords of a request definition, cause the next request to be implicitly scheduled.

The remote queuing facility of BCOM enhances your queuing capabilities by defining a new type of request called the Remote Queue (RQ); you can specify it as the object of either form of request queuing commands:

- you can specify a Remote Queue request name as the object of a QUEUE command;
- or you can specify a Remote Queue request name as the NORMAL-END and ABNORMAL-END keywords of a request definition: this will implicitly schedule the remote queue request based on the completion status of the ancestor requestor.

The Remote Queue Request

A Remote Queue request is like any other request definition that you add to the request file of your local BCOM location; the difference is that its remote file name is actually the name of a request that exists at the request's remote location.

However, since you specify that remote request name inside a local request - under the same or different name - you can maintain locally a record of what is queued remotely, you can change the remote request name without having to change local requests that reference it. Your remote queue request is always under control because you can keep track of its scheduling and progress using the Queue management commands of BFSINT, and the BCOM logs at both ends provide you with a detailed trace of its execution.

How to define a Remote Queue Request

Defined as TYPE=RQ, a Remote Queue request is easy to build:

- because it is queued like any other request, the Remote Queue request comprises standard BCOM scheduling keywords
- because it must be routed to a remote location, the Remote Queue request also comprises standard BCOM routing keywords
- and because the resource to be used at the remote location is in fact a predefined BCOM request, the Remote Queue request uses the remote-filename keyword to specify the desired request name.

Sample definition

```
SET TYPE            RQ
SET LOCAL-LOCATION  %TAN1
SET REMOTE-LOCATION %MVS1
SET REMOTE-FILENAME #EFGH
SET ROUTING-CLASS   1
SET PRIORITY-CLASS  5
SET COMMENT         queue request #EFGH at %MVS1
ADD #ABCD
```

The above sample request will be queued at priority 5 to the routing class 1 of the local BCOM program at %TAN1. When it runs, the request will cause that local BCOM to send a remote queue command to the BCOM program running at the remote location %MVS1; this remote queue command will initiate request #EFGH, found in the request file of that given BCOM location.

Alternate way of Defining the Request Name

There is an alternate way of providing the name of the request to be scheduled at the remote location: if you omit the REMOTE-FILE keyword from your request definition, then the local BCOM program will assume that the request to be queued at the remote location has the same name as your local request definition (e.g. #ABCD, in this case).

Example of how to use RQF

To clarify these concepts, let us show you with an example, how you can distribute a file from an MVS corporate mainframe to a set of different BCOM locations, using the Remote Queuing Facility of BCOM.

Let us begin with a sample network consisting of a central MVS platform and a NSK Guardian node. Connected to the NSK Guardian is a Windows NT platform.

To take into consideration the timeliness of network connections between these platforms, our distribution procedure will be hierarchical: BCOM-MVS will be responsible for distributing the corporate file to its adjacent nodes, and the BCOM program on the latter nodes will be responsible for distributing the same file to their own adjacent platforms.

To complete the picture, let us suppose that this distribution must be initiated on the 25th of October 1992, at 3 PM, and every Sunday thereafter.

The MVS Request File

In our sample network, the BCOM Request File on MVS will contain the following requests:

- #SNDFNSK - a SEND-FILE to distribute the corporate file to the adjacent NSK Guardian node; its NORMAL-END then triggers the local request #SNDFWT
- #SNDFWT - a REMOTE-QUEUE request to queue a request having the same name in the request file of the adjacent NSK Guardian node.

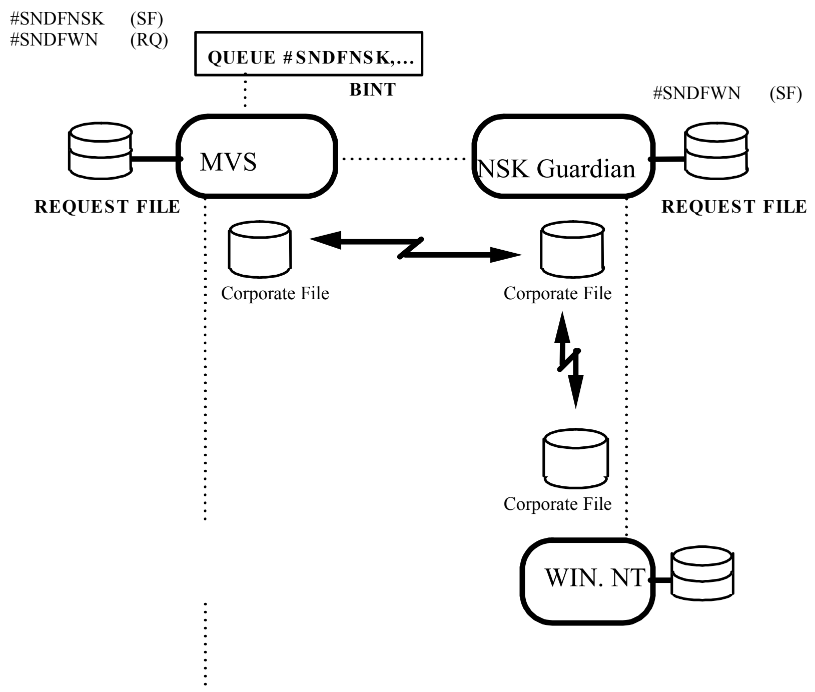
The NSK Guardian Request File

The BCOM Request File on NSK Guardian will contain the following request:

- #SNDFWN - a SEND-FILE to distribute the corporate file to the adjacent Windows NT node

In our examples, we have purposely left out certain functions (like the ABNORMAL-END) for the sake of clarity and simplicity.

Network topology for distribution example



Initiating the Distribution

Back on the MVS platform, the entire distribution can be queued for scheduling by using the following command, thanks to follow-on scheduling:

```
   QUEUE #SNDFT16(01,05,9210251500,007)
```

Security for the Remote Queue Facility

Local Platform Security

The ID of the user that queues a remote queue request is always subject to the command authority control of BCOM: i.e. the user must be authorized for issuing a QUEUE or QWAIT command.

Similarly, the user must be authorized to issue any other QUEUE management command against a Remote Queue request in progress (e.g. QHOLD, QDEL, QREL, etc.).

No other security checks are required for handling a Remote Queue request since it does not access resources on the initiating platform.

Remote Platform Security

BCOM always propagates the ID of the user that queues a request: (1) to the BCOM server at the receiving remote location; and (2) to the local child requests that are triggered from this ancestor when the keywords of follow-on scheduling are being used.

When a user-ID is shipped to a remote location configured for using BCOM security (e.g. with a security level greater than 0), the BCOM server at that remote location correlates this name to a local user-id valid for that receiving platform. The Remote Queue request name is then queued under that correlated ID, into the receiving BCOM Queue file.

Security Return Codes

Since a Remote Queue request will cause userid translation to take place and a queue command to be executed at the remote location, the following error codes may be returned:

- security violation - userid cannot be correlated or userid not authorized for queue command
- function (e.g. accept remote queue requests) is not supported
- remote request name is not found in the request file at that remote BCOM location
- remote request specified is already scheduled or queued for execution at that remote BCOM location
- remote request was queued successfully

Remote queue considerations

A Remote Queue request is ultimately a method for issuing a QUEUE command remotely: as with any QUEUE command (except for QWAIT), please remember that a successful queuing operation does not necessarily mean a successful execution. This is one of the reasons why our example suggests using a hierarchical (or cooperative) method for distributing the corporate file.

Follow-on scheduling

This section discusses how to request Automatic Conditional Request Chaining, a facility of BCOM for automating transfers and jobs in distributed environments. As such, it illustrates the basic steps in planning and defining conditional request scheduling, and provides a complete example illustrating the flexibility and ease of use of this powerful facility.

The section is organized according to the following subsections:

1. Overview of Automatic Follow-on Scheduling explains what this facility is all about, the basic principles of operation and how to specify it in your request definitions. to organize a stream of requests chained through their normal or abnormal termination status.

2. Example of Automatic Follow-on Scheduling illustrates with the help of one complete example, how to organize a stream of requests chained through their normal or abnormal termination status.

Overview

Request scheduling does not have to involve the presence of an operator. By managing the control files with the help of the Automatic Scheduler Facility, BCOM will allow predefined requests to be queued for execution based on the outcome of previous transfers.

When the Automatic Scheduling function is used in conjunction with BCOM's Job Submission facility, then serial batch job processing can be effected across BCOM-supported platforms. This eliminates the duplication of files across the environments, and allows for the processing of data files at the locations where they will most often be used. Triggering of the 'follow-on' requests is completely automated by the BCOM monitor and is dependent solely on the outcome of the ancestor request—that is normal or abnormal termination.

When a BCOM request is defined via the BFSINT, you may specify the name of another request to be started if the current request completes normally. Similarly, a request name may be started if the current request ends abnormally.

Normal end definition

A request execution is recorded successful if:

- the remote server transmits a message stating the normal end of processing
- the local io-processes associated to the transfer end without abend and returns a completion code present in the list of the NORMAL-IO-COMPLETION-CODES.

An IO process is started at each request execution to process the data exchanged with the remote server. The IO process is can be a Guardian utility (FUP, BACKUP, RESTORE, SQLCI, TACL) or a program part of BCOM (BFSCPYX, BFSLOAD, BFSBKUP, BFSREST). A backup or restore request involves two IO processes: BFSBKUP and BACKUP or BFSREST and RESTORE. When two IO processes execute, the completion code of both is checked against the list of acceptable codes.

The list of acceptable completion codes is configured globally by the keyword NORMAL-IO-COMPLETION-CODES, and can be overridden at request level by the local attribute with the same keyword and syntax.

Without any setting, the default list of accepted completion code is:

```
NORMAL-IO-COMPLETION-CODE 0,1,8
```

which can result in unsuccessfully execution in BCOM V5.x of a request that was successful in a pre-V5.0 BCOM

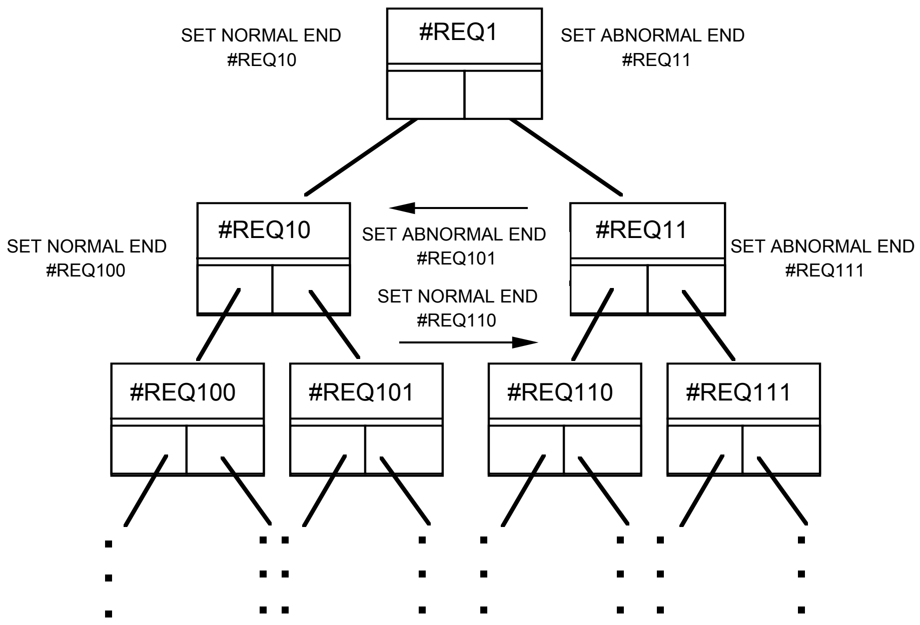
Note that the completion code returned to BCOM by LJ or RJ requests executing a TACL obey files is the completion code of the TACL process, not the completion code of the user application started by TACL.

Request Definition

Each request definition can be seen as the root of a binary tree - with the abnormal end sub trees generated on one side and the normal-end sub trees generated on the other side.

It is perfectly legal to have circular chaining, although one must take care that this does not create a runaway task or looping situation.

The conditional chaining takes place at the location from which the transfer is initiated - even though the actual processing is taking place at a remote location. The termination status is returned to the requesting platform and used to select the next request, provided conditional chaining has been specified.



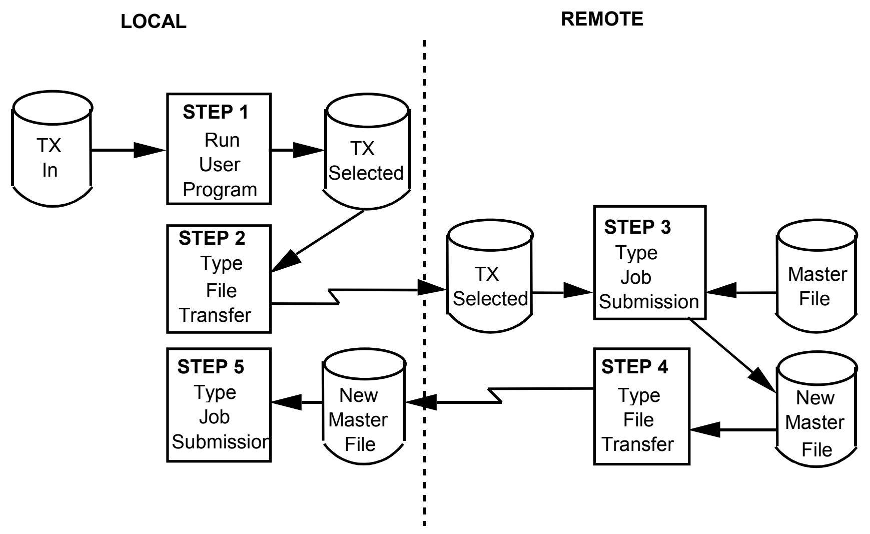
Note, however, that a job submission request can open the door to conditional request chaining performed at the remote location as well. This requires access to the request file of the remote platform, through the standard BFSINT program.

For example, a job submitted to a remote platform may include a step in which the local BFSINT - i.e. at that remote location - is invoked in order to start, define or purge new or existing requests. Moreover, such a JOB step may be executed conditionally, based on JCL return codes. In that case, two separate tree roots are begun, each one active on its own platform.

To clarify these concepts and illustrate the flexibility of Conditional Request Chaining, we now propose a simple job stream, which executes on two separate platforms.

Example of Conditional Request Scheduling

The example illustrates how job scheduling can be used for distributed batch processing across heterogeneous environments.


LOCAL
REMOTE
TX In
STEP 1 Run User Program
TX Selected
STEP 2 Type File Transfer
TX Selected
STEP 3 Type Job Submission
Master File
STEP 4 Type File Transfer
New Master File
STEP 5 Type Job Submission
New Master File

Step 1

On the local platform, is an Obey file which runs the user program that creates the file of selected transactions. The Obey ends by calling the BFSINT and queuing step 2.

```
COMMENT STEP 1 - SELECTION
COMMENT
ASSIGN TX-IN, $DATA01.DATA.TXIN
ASSIGN TX,    $DATA01.DATA.TXSEL
RUN <STEP 1 PROGRAM>

COMMENT STEP 2 -FILE TRANSFER TO MVS USING BCOM
COMMENT
RUN BFSINT $BMONX;QUEUE #STEP2
```

Step 2

On the local platform, is a BCOM SEND-FILE request that transfers the transaction file to the MVS system. If that transfer is successful, then BCOM will automatically initiate step 3.

```
SET TYPE            SF
SET LOCAL-LOCATION  %TAND1
SET REMOTE-LOCATION %MVSMTL
SET ROUTING-CLASS   2
SET LOCAL-FILE      $DATA01.DATA.TXIN
SET LOCAL-ATTR01    PROCESS-PRIORITY=100
SET LOCAL-ATTR02    REPORT OFF
SET REMOTE-FILE     DATA01.TXSEL
SET REMOTE-ATTR01   UNIT=SYSDA
SET REMOTE-ATTR02   DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
SET REMOTE-ATTR03   DISP=(NEW,CATLG,DELETE)
SET REMOTE-ATTR04   SPACE=(TRK,(5,5))
SET NORMAL-END      #STEP3
SET ABNORMAL-END    #ALERT
ADD #STEP2
```

If the transfer completes successfully, the request #STEP3 will be queued; otherwise, the #ALERT request will be queued. The #ALERT request can submit a JCL or Obey file to notify the abnormal-end of the job stream.

Step 3

on the local platform, is a BCOM REMOTE-JOB request that sends JCL stored on the local platform to the MVS internal reader. This JCL defines a job for processing the transaction file against the new master.

```
SET TYPE            RJ
SET LOCAL-LOCATION  %TAND1
SET REMOTE-LOCATION %MVSMTL
SET ROUTING-CLASS   2
SET LOCAL-FILE      $DATA01.JCL.STEP3
SET LOCAL-ATTR01    PROCESS PRIORITY = 100
SET LOCAL-ATTR02    REPORT OFF
RESET NORMAL-END
SET ABNORMAL-END    #ALERT
FIELD NONE
```

The job stream defined by that JCL updates the master file with the file of selected transactions in order to create a new master file. If the new master file is created successfully, then two additional steps of the JCL are executed:

(1) STEPXX invokes the B62BINT to initiate a file transfer request which will move the new master file back to the original platform.

(2) When that transfer completes normally, a remote job submission request will be triggered that will cause this new master file to be processed at the original platform location.

```
//<your job> JOB...
//...STEP TO WRITE NEW MASTER FILE
   .
   .
//STEPXX   EXEC PGM=B62BINT,
//         PARM='xxxxxxxx,yyyyyyyy',
//         COND = (4,LT)
//STEPLIB  DD   DISP=SHR,DSN=YOUR.MVS.library
//COMMOUT  DD   SYSOUT = A
//SYSPRINT DD   SYSOUT = A
//SYSOUT   DD   SYSOUT = A
//COMMIN   DD   *
QUEUEWAIT #STEP4
/*
```

where:

xxxxxxxx is equal to the APPLID of the B62BINT;

yyyyyyyy is equal to the APPLID of BCOM.

Step 4

On the remote platform is used to carry out the following file transfer (remote to local).

```
SET TYPE            SF
SET LOCAL-LOCATION  %MVSMTL
SET REMOTE-LOCATION %TAND1
SET LOCAL-FILE      DATA01.MASTER1
SET REMOTE-FILE     $DATA01.DATA.NMASTER
SET LOCAL-ATTR01    DISP=SHR
SET REMOTE-ATTR01   PROCESS-PRIORITY = 100
SET REMOTE ATTR02   REPORT  OFF
SET NORMAL-END      #STEP5
SET ABNORMAL-END    #ALERT
ADD #STEP4
```

If this transfer completes successfully, then the request #STEP5 will be executed.

Step 5

also on the remote platform, is used to cause a remote job submission to occur on the NSK Guardian; in that JOB, a TACL command file will process the newly transferred master file.

```
SET   TYPE            RJ
SET   LOCAL-LOCATION  %MVSMTL
SET   REMOTE-LOCATION %TAND1
SET   REMOTE-FILE $   DATA01.TACL.STEP5
SET   REMOTE-ATTR1    OUTFILE $S.#LPA
SET   REMOTE-ATTR2    PROCESS-PRIORITY = 100
SET   REMOTE-ATTR3    JOB-WAITED = YES
SET   ROUTING-CLASS   02
SET   PRIORITY-CLASS  05
RESET NORMAL-END
SET   ABNORMAL-END #ALERT
ADD   #STEP5
```

BCOM error recovery

Error recovery is installation and/or application dependent and can be controlled though the programmatic interface, as desired.

There are a variety of reasons for recovery, and several recovery methods may be applied, as follows:

- Display to the console
- Resubmit the transfer
- Change transmission routes from Follow-on Scheduling
- Activate abnormal conditions from Follow-on Scheduling
- Trap completion notification messages sent to EMS
- Process completion notification messages in job submission

Data can be supplied to BCOM by means of user-written I/O processes. These user-written I/O processes can replace the conventional disk file data transfer approach when more interactive or real-time data shipments across platforms are desired.

The purpose of this section is to discuss various facilities of the BCOM product that let you handle possible errors and recover from data transfer failures.

The facilities discussed include: retries, routing-classes, auto-restart, checkpoint-restart, abnormal-end notification and follow-on scheduling.

When data transfers take place between heterogeneous platforms they can encounter various hazards due simply to the great quantity and diversity of hardware, software and environments that are involved in the interconnections.

BCOM is designed to take some of these hazards into consideration and manage them for you; for other situations, specific features have been developed that will help you address the potential for abnormal transfer completion: these features range from configuration alternatives to request definition options.

Communication retry capabilities

Retries

In the configuration file of BCOM, you can define the number of times that a failed operation or a Logical Unit or session in error will be retried. The range and the default values for these are defined in the installation manual of your platform.

Of course, this parameter is designed to take care of errors that can be retried - but especially, those errors that come from establishing sessions or re-establishing them after a communication failure.

Retry Interval

As long as the limit has not been reached, BCOM will retry the failed operation. But sometimes, there is no point in retrying an operation immediately; time is needed to restore the needed or failed resources to a useable state.

Between automatic retries, BCOM will wait the prescribed amount of time that you define - expressed in minutes - in the configuration file of BCOM. The range and default values are found in the installation manual of your platform. Let's see how you can make use of this information.

Suppose that the majority of your operations that can be retried come from establishing sessions with a remote BCOM, which is not under your control. In that case, you may want to configure your local BCOM in the following manner:

- define that remote location with an AUTOSTART attribute: BCOM will try to start that connection as soon as it starts up and following any connection failure, thereafter
- define a large value for the number of retries
- specify a retry interval of 10 minutes between attempts.

Suppose that BCOM has just completed another unsuccessful attempt at re-establishing the connection to your remote location when you are informed that the communication resources formally unavailable are now free to connect. You won't have to wait for another 10 minutes: you can always override a retry interval by immediately entering a WARMSTART command for the designated Logical Unit(s).

Defining alternate paths

What we are about to discuss, here, is the way to set up the communication paths between BCOM platforms and the flexibility that the configuration file of BCOM provides in letting you set up alternative ways of reaching the same destination. But first, let us review briefly the concepts of the BCOM routing class.

With BCOM, you can use multiple Logical Units (LUs) to exchange data between two BCOM locations; you can group these LUs under a unique PROCID - remember that on an MVS or NSK Guardian platform, requestor and server processes (or PROCIDs) can multithread several LUs simultaneously - or you can define them under different PROCIDs.

Next, each PROCID is assigned to a Routing Class. A routing class can use one or more PROCIDs to service the load of data transfer requests. The Routing Class type determines whether these transfers will be done in standard or multiplex fashion. Routing Classes are identified by a unique routing class value.

Finally, the Routing Class is linked to a specific Remote Location; one or more routing classes can be defined to connect to the same remote location.

Now, consider that Logical Units refer to a specific physical medium, which means that by using different routing classes in your request definitions, you can effectively select different communication paths going to the same remote location. For example, a NSK Guardian-MVS remote-location connection could define one routing class going through an NCP, another one going over a SNAXLINK channel connection and a last one using NSK Guardian's FOX and an external X25PAD on MVS.

When you initiate a request (with the Queue command), you can even override a predefined routing class and select a different path for your data transfer. Thus routing classes can be used as a means to define alternate or backup paths for your data transfers.

Auto-restart

AUTO-RESTART is a request definition feature that restarts automatically a request that failed because of a communication error. BCOM keeps the failed request into the queue file until the communication resources are once again available and then to automatically restart the request from the beginning or last checkpoint.

Here is how you specify this feature in your request definition; note that the default value for the AUTO-RESTART keyword is "NO".

```
SET TYPE          SF
SET ROUTING-CLASS 2
SET AUTO-RESTART  YES
•••
```

If you display the items in the queue while your request is to be restarted, you will see that BCOM assigns it a special status of "waiting for auto-restart". Impact of re-execution: You should take into consideration the possible impact at the remote location of restarting a data transfer - especially the impact of MVS file disposition (JCL parameter DISP=) that is executed a second time.

If the request is executed without checkpoint (CHECKPOINT 0), the second restarted execution is handled as a distinct, brand new, execution. The dispositions NEW, MOD are likely to make the request fail or create invalid results. DISP=OLD (or DISP=SHR) does not change the status of the file and allows re-execution. If a file has to be created by the execution of a request, DISP=(SCR,CATLG,DELETE) will allow re-executions.

If the request is executed with checkpoint, the disposition and generation +1 of a GDG are interpreted by BCOM at restart time, as presented in Checkpoint/restart section below.

Handling MVS GDGs

The MVS platform supports a special type of dataset called the Generation Dataset Group (GDG), which may come in handy with AUTO-RESTART issues. To understand why, first, let us explain what a GDG is.

A GDG is a type of dataset partitioned for a limited set of reusable members. These members are addressable through a special notation convention: the suffix (+1) means "add the data as the next generation"; (-1) refers to the previous generation; (0) refers to the current generation. Each time you write with an index of (+1), the system uses the next member, in a wraparound fashion. Internally, the system assigns a unique fully qualified name to each "generation" of the GDG, but on your JCL, you only specify "(+1)".

Many installations use a GDG for small remote file backups: they define a 7-member GDG and backup their daily data to each member—using an index of (+1).

Now, what would happen if BCOM were to restart a transfer to such a dataset? Unless something special is done, BCOM would be writing today's data into "tomorrow's generation" member - which would not be the desired outcome.

But something special is indeed being done with BCOM, so that you can use MVS GDGs without fear: BCOM will handle an AUTO-RESTART request by relocating automatically to the most current (at the time of the failure) generation created.

Checkpoint-restart

CHECKPOINT-RESTART is a different feature, designed to recover from long running data transfers. BCOM registers regularly checkpoints along the transfers, and if a request fails, the transfer will be restarted from the last valid checkpoint. When checkpoint is enabled, the failed request will be held in the queue of requests to executed, - whatever the reason of failure. A manual intervention is required (after fixing the reason of error) to restart the request: QRESTART #reqname.

If the request was also defined with Auto-restart Yes, and the reason of error is a communication error, the request will be automatically restarted.

The checkpoint/restart feature is enabled by a non-zero value of CHECKPOINT parameter of the request. The value of CHECKPOINT determines the checkpoint interval.

```
SET TYPE          SF
SET ROUTING-CLASS 2
SET CHECKPOINT    nnn
•••
```

The value specified for checkpoint interval (nnn) represents a record counter for file transfers, and a number of megabytes for backup requests:

- the smaller the interval, the smaller the quantity of data that has to be retransmitted - but at the same time, more checkpoints have to be taken.
- the greater the interval, the greater the quantity of data that has to be retransmitted - but at the same time, fewer checkpoints have to be taken.

For a file-transfer, this interval value should be ideally a multiple of the WINDOW-COUNT in the configuration file of BCOM.

For a backup transfer, this interval should be quite large, a value larger than the minimal 250 MB is suggested.

While the request is sitting in the queue, BCOM will assign it a special status of "waiting for manual restart" or waiting for AUTO-RESTART.

Impact of re-execution

The MVS file disposition (JCL parameter DISP=) can be an inconvenient when a transfer is restarted, making the DISP parameter executed a second time.

To help manage the DISP parameter, BCOM imposes some restrictions and interpret/modify the DISP and the GDG generation +1 at restart time, avoiding manual modification of the request to allow restart.

Restrictions:

- DISP=SCR, or DISP=MOD are forbidden
- to create a file on MVS side, the only disposition supported is: DISP=(NEW,CATLG,CATLG)

Interpretation of request definition at restart time:

- if the disposition is DISP=(NEW,…), the file is supposed already created and catalogued, the disposition is internally replaced by DISP=OLD
- if the MVS file name ends by “(+1)”, the end of file name is internally replaced by “(0)”

When to Use

The CHECKPOINT-RESTART facility is used to recover from failures involving very large files.

There are certain considerations for using the checkpoint-restart facility:

- this facility is available only for transfer of structured files (e.g. not for NSK Guardian unstructured files) and for backup requests.
- If AUTO-RESTART = NO, then you must manually restart the transfer, using the QRESTART command;
- the QRESTART command allows you to override the checkpoint value and specify a restart "from the beginning". but the MVS disposition and generation +1 are still interpreted as in other restart cases.

I/O process conventions

Using a process to send or receive data

When, a transfer from NSK Guardian to the remote partner is initiated, BCOM (BCOPYX) opens the process and issues a prompt (message with length = 0) to the process. This process can then pass records using the GUARDIAN reply procedure (through its $receive). To indicate end of file, the process returns a reply code of 1.

When a transfer to NSK GUARDIAN from the remote partner is initiated, BCOM (BLOAD) opens the process, writes the data record to the opened process, which receives the records through its "$RECEIVE" file. The process replies to BLOAD by using the GUARDIAN REPLY procedure call (through $receive). To indicate end of file, BLOAD will close the process.

The following pseudo-code demonstrates how to code a process for from and to NSK Guardian file transfers.

From NSK Guardian Transfer

```
CALL OPEN ($RECEIVE...)
CALL READUPDATE ($RECEIVE   )     ! Read Startup message
CALL REPLY
WHILE NOT END^OF^LOOP DO
  BEGIN
    CALL READUPDATE $RECEIVE
    IF there is a record to be sent THEN
      CALL REPLY (record to send and reply code = 0)
     ELSE
      BEGIN
        CALL REPLY (reply code = 1)
        END^OF^LOOP = TRUE
      END
    END
```

To NSK Guardian Transfer

```
CALL OPEN ($RECEIVE...)
CALL READUPDATE ($RECEIVE   )     ! Read Startup message
CALL REPLY
WHILE NOT CLOSE^MESSAGE DO
  BEGIN
   CALL READUPDATE $RECEIVE
  .!.process record just read
   CALL REPLY (Reply code = 0)
  END
```

I/O Process Notes

When using the I/O PROCESS programmatic interface to BCOM, a request must still be set up and queued as in conventional file transfers. Simply replace the disk file name with the user-supplied process name that will be receiving or supplying the data to the BCOM I/O processes (i.e. BCOPYX for transfer from NSK Guardian, or BLOAD for transfer to NSK Guardian).

An example of a request setup for a process driven transfer would be:

```
SET TYPE            SF
SET LOCAL-LOCATION  %TAND1
SET REMOTE-LOCATION %MVSMTL
SET ROUTING-CLASS   2
SET PRIORITY-CLASS  1
SET LOCAL-ATTR01    PROCESS-PRIORITY = 190
SET LOCAL-FILE      $USR
SET REMOTE-FILE     MVS.TARGET.DATASET
SET REMOTE-ATTR01   DISP=SHR
ADD #PROREQ
```

The process $USR does not have to exist at the time the request is added, but must be running at the time the request is queued.

Note: *The user may also wish to queue the request programmatically through the programmatic interface BFSINT; facility. See the previous section.*

This would make the entire transfer operation "process-controlled."

The following subsection illustrates two fully coded program examples for both sending and receiving processes.

```
CURRENT FILE IS $DATA01.BCOMTEST.TRECEIVS

?INSPECT, SYMBOLS
?NOLIST
?SOURCE $SYSTEM.SYSTEM.GPLDEFS
?LIST
!
!  TAL  -  TRECEIVS
!  This sample program reads the records sent by
!  BFSLOAD through $RECEIVE
!  and writes the records to the out file.
!  The user specifies the out file with the
!  parameter OUT in the RUN statement.
!   (BFSLOAD transfer records from MVS to NSK GUARDIAN)


LITERAL ^EOF           = 1,
        ^OPEN^MSGID    = -30,
        ^CLOSE^MSGID   = -31,
        ^STARTUP^MSGID = -1;

INT      REPLY^STATUS,
         PROCESS^ID[0:3],
        .BUFFER    [0:9999],
         READ^COUNT,
         ERROR^NUM,
         RECEIVE[0:11] := ["$RECEIVE", 8 * ["  "]],
         RECEIVE^FN,
         STMSGOUT^FN,
        ^NB^OPEN := 0;

STRUCT   STARTUP^MSG^DEF (*);
   BEGIN
   INT      MSGID;
   INT      DEFAULT[0:7];
   INT      IN [0:11];
   INT      OUT[0:11];
   STRING   TXT[0:527];
   END;

INT     .STMSG (STARTUP^MSG^DEF) := @BUFFER;

?NOLIST
?SOURCE $SYSTEM.SYSTEM.EXTDECS0
?LIST

?PAGE "MAIN PROCEDURE"
?SECTION MAIN^PROC
!**************************************************************
!*                                                            *
!**************************************************************
PROC MAIN^PROC MAIN;

BEGIN

  CALL OPEN (RECEIVE, RECEIVE^FN, %40000, 1);
  IF <> THEN
    CALL DEBUG;

  CALL READUPDATE (RECEIVE^FN, BUFFER, 10000, READ^COUNT);
  IF < THEN
    CALL DEBUG;

  WHILE BUFFER <> ^CLOSE^MSGID DO
    BEGIN
      REPLY^STATUS := 0;
      CASE $ABS (BUFFER) OF
        BEGIN
        ! 0 !   ;
        ! 1 !   BEGIN
```

```
                  REPLY^STATUS := 0;
                END;
        ! 2 !   ;
        ! 3 !   ;
        OTHERWISE;
        END;

      CALL REPLY (,,,, REPLY^STATUS);
      IF < THEN
        CALL DEBUG;

      CALL READUPDATE (RECEIVE^FN, BUFFER, 10000, READ^COUNT);
    END;

  CALL REPLY (,,,, 0);
  IF < THEN
    CALL DEBUG;

  CALL OPEN (STMSG.OUT, STMSGOUT^FN);
  IF <> THEN
    CALL DEBUG;

  WHILE 1 DO
    BEGIN
      CALL READUPDATE (RECEIVE^FN, BUFFER, 10000, READ^COUNT);
      IF > THEN          ! SYSTEM MESSAGE
        BEGIN
          IF BUFFER[0] = ^OPEN^MSGID THEN
            BEGIN
              IF BUFFER[0] = ^OPEN^MSGID THEN
                ^NB^OPEN := ^NB^OPEN + 1;
            END
           ELSE IF BUFFER[0] = ^CLOSE^MSGID THEN
            BEGIN
              ^NB^OPEN := ^NB^OPEN - 1;
              IF ^NB^OPEN = 0 THEN
                CALL  STOP;
            END;
        END
       ELSE IF < THEN    ! ERROR
        BEGIN
          CALL FILEINFO (RECEIVE^FN, ERROR^NUM);
          CALL DEBUG;
        END
       ELSE              ! DATA
        BEGIN
          CALL WRITE (STMSGOUT^FN, BUFFER, READ^COUNT);
          IF < THEN
            BEGIN
              CALL FILEINFO (STMSGOUT^FN, ERROR^NUM);
              CALL DEBUG;
            END
        END;

      CALL REPLY (,,,, 0);
    END;

END;   ! MAIN^PROC
```

```
CURRENT FILE IS $DATA01.BCOMTEST.TSENDERS

?INSPECT, SYMBOLS
?NOLIST
?SOURCE $SYSTEM.SYSTEM.GPLDEFS
?LIST
!
!  TAL  -  TSENDERS
!  This sample program reads from the in file (file, process or
!  terminal) supplied by the user with the "IN" parameter of
!  the run command.  Records are read from the in file and sent
!  to the BCOPYX module.
!
!  This is accomplished with the GUARDIAN reply procedure.
!
!  (BCOPYX transfer records from NSK GUARDIAN to
!   Remote-Partner)

LITERAL ^OPEN^MSGID    = -30,
        ^CLOSE^MSGID   = -31,
        ^STARTUP^MSGID = -1,
        ^REPLY^STARTUP = 70,
        ^BLKBUFLEN     = 1024,
        ^PROMPT        = "@";
INT     .BLKBUF[0:511],
         IN^FILE   [0:FCBSIZE-1],
         COMMON^FCB[0:FCBSIZE-1] := 0,    !  MUST BE ZERO HERE
         REPLY^STATUS,
         PROCES^ID[0:3],
         TAG       [0:1],
        .BUFFER    [0:9999],
        .RECORD    [0:2047],
         READ^COUNT,
         ERROR^NUM,
         RECEIVE[0:11] := ["$RECEIVE", 8 * ["  "]],
         RECEIVE^FN,
         STMSGOUT^FN,
         STMSGIN^FN;
STRING  .SBUF    := @BUFFER '<<' 1,
        .SRECORD := @RECORD '<<' 1;

STRUCT   STARTUP^MSG^DEF (*);
   BEGIN
   INT      MSGID;
   INT      DEFAULT[0:7];
   INT      IN [0:11];
   INT      OUT[0:11];
   STRING   TXT[0:527];
   END;

INT     .STMSG (STARTUP^MSG^DEF) := @BUFFER;

?NOLIST
?SOURCE $SYSTEM.SYSTEM.EXTDECS0
?LIST

?PAGE "PROCESS^BCPY^RQST"
?SECTION PROCESS^BCPY^RQST
!**************************************************************
!*                                                            *
!**************************************************************
PROC PROCESS^BCPY^RQST (BUFFER, LEN);
INT     .BUFFER,
         LEN;

BEGIN
INT      COUNT^READ;

  CALL LASTRECEIVE       (PROCES^ID, TAG);
  ERROR^NUM := READ^FILE (IN^FILE, RECORD, COUNT^READ);
  IF ERROR^NUM THEN
```

```
        BEGIN
          IF ERROR^NUM = 1 THEN           ! EOF
            CALL REPLY (,,, TAG, 1)
           ELSE IF ERROR^NUM = 111 THEN   ! BREAK AT TERMINAL, WE STOP
            CALL STOP
           ELSE CALL DEBUG;
          RETURN;
        END;

      CALL REPLY (RECORD, COUNT^READ,, TAG, REPLY^STATUS);
      IF <> THEN
        CALL DEBUG;

    END;   ! PROCESS^BCPY^RQST


    ?PAGE "MAIN PROCEDURE"
    ?SECTION MAIN^PROC
    !***************************************************************
    !*                                                             *
    !***************************************************************
    PROC MAIN^PROC MAIN;

    BEGIN

      CALL OPEN (RECEIVE, RECEIVE^FN, %40000, 1);
      IF <> THEN
        CALL DEBUG;

      CALL READUPDATE (RECEIVE^FN, BUFFER, 10000, READ^COUNT);
      IF < THEN
        CALL DEBUG;

      WHILE BUFFER <> ^CLOSE^MSGID DO
        BEGIN
          REPLY^STATUS := 0;
          CASE $ABS (BUFFER) OF
            BEGIN
            ! 0 !   ;
            ! 1 !   BEGIN
                      REPLY^STATUS := 0;
                    END;
            ! 2 !   ;
            ! 3 !   ;
            OTHERWISE;
            END;

          CALL REPLY (,,,, REPLY^STATUS);
          IF < THEN
            CALL DEBUG;

          CALL READUPDATE (RECEIVE^FN, BUFFER, 10000, READ^COUNT);
        END;

      CALL REPLY (,,,, 0);
      IF < THEN
        CALL DEBUG;

      CALL OPEN (STMSG.OUT, STMSGOUT^FN);
      IF <> THEN
        CALL DEBUG;

    !
    !       SIO PROCEDURES FOR FILE AND TERMINAL ...
    !
      CALL SET^FILE (COMMON^FCB, INIT^FILEFCB);

      CALL SET^FILE (IN^FILE, INIT^FILEFCB);
      CALL SET^FILE (IN^FILE, ASSIGN^FILENAME,   @STMSG.IN);
      CALL SET^FILE (IN^FILE, ASSIGN^OPENACCESS, READ^ACCESS);
      ERROR^NUM := OPEN^FILE (COMMON^FCB, IN^FILE, BLKBUF, ^BLKBUFLEN,,,
                              200, ^PROMPT);
      IF ERROR^NUM THEN
        CALL DEBUG;

      WHILE 1 DO
        BEGIN
          CALL READUPDATE (RECEIVE^FN, BUFFER, 10000, READ^COUNT);
          IF > THEN
            BEGIN
              IF BUFFER [0] = ^OPEN^MSGID THEN
                CALL REPLY (,,,, 0)
               ELSE IF BUFFER[0] = ^CLOSE^MSGID THEN
                CALL STOP
               ELSE CALL REPLY (,,,, 0);
            END
           ELSE IF < THEN
            BEGIN
              CALL FILEINFO (RECEIVE^FN, ERROR^NUM);
              CALL DEBUG;
            END
           ELSE CALL PROCESS^BCPY^RQST (BUFFER, READ^COUNT);
       END;

    END;   ! MAIN^PROC
```

Using a process to send blocked data — VARIN option

The following describes the VARIN option (which is modeled after the FUP feature of the same name) for sending blocked records to a remote BCOM location from a user process.

Request Definition

The request to have a NSK Guardian user process send blocked data to BCOM can be initiated by the local NSK Guardian (e.g. SF) or by a remote BCOM location (e.g. RF).

In either case, to use the VARIN feature of BCOM, the request definition must meet certain requirements:

- a remote RF request must reference the NSK Guardian user process as a remote file; a local SF must reference it as the local file;
- a remote RF request must include the remote-attribute PARMS=VARIN; a local SF request must include the local-attribute PARMS=VARIN

The AUTORESTART and CHECKPOINT request options must NOT be used.

Example 1

Local SF (request initiated by the NSK Guardian platform):

```
SET TYPE            SF
SET LOCAL-LOCATION  %TND2
SET LOCAL-FILE      $<USER.PROCESS.NAME>
SET LOCAL-ATTR01    PARAMS=VARIN
SET REMOTE-LOCATION %MVSMTL
SET REMOTE-FILE     <MVS.TARGET.FILE>
SET REMOTE-ATTR01   DISP=SHR
SET ROUTING-CLASS   02
SET AUTO-RESTART    NO
SET CHECKPOINT      0
SET COMPRESSION     NO

ADD #RQSTNO1
```

<u>Example 2</u>

Remote RF (request initiated by a remote platform):

```
SET TYPE            RF
SET LOCAL-LOCATION  %MVSMTL
SET LOCAL-FILE      <MVS.TARGET.FILE>
SET LOCAL-ATTR01    DISP=SHR
SET REMOTE-LOCATION %TND2
SET REMOTE-FILE     $<USER.PROCESS.NAME>
SET REMOTE-ATTR01   PARAMS=VARIN
SET ROUTING-CLASS   02
SET AUTO-RESTART    NO
SET CHECKPOINT      0
SET COMPRESSION     NO

ADD #RQSTNO2
```

Process Specification

Your process must be running by the time that the SF BCOM request definition executes, or the remote RF is received and processed by BCOM. BCOM will open your process and read blocks of data from it.

These operations are executed in the following order:

I. Your process must accept BCOM's open message on the $RECEIVE file.

II. Your process must accept the WRITEREAD requests from BCOM: those take the form of a zero-length WRITE/non-zero length READ completion on your $RECEIVE file.

III. Your process must respond by writing a block of data, built according to the following specifications:

- the IPC buffer has a maximum size of 4074 bytes;
- this buffer contains variable length records;
- each record begins with a 2-byte length field, followed by the data;
- your application can send either variable length blocks or fixed length blocks; if it sends variable length blocks, then there is no need to add end-of-data indicators, as all blocks will be exactly filled by data and BCOM will be aware of the length of each block, through the IPC reply size. On the contrary, if your application sends fixed length blocks, then the end of data in each block must be signalled by the addition of a word-aligned (see below) 2-byte end-of-data indicator containing -1 (this does not apply if the data ends exactly, or one byte before the end of the block);

- the length field words and the end-of-data word must be aligned on even byte boundaries (a pad space has to be added to the previous record if necessary).

IV. When it has nothing more to send to BCOM, your process must issue a call to REPLY, specifying an error code of 1, which is the NSK Guardian convention for end of file.

V. Your process will receive BCOM's close message.

Notes:

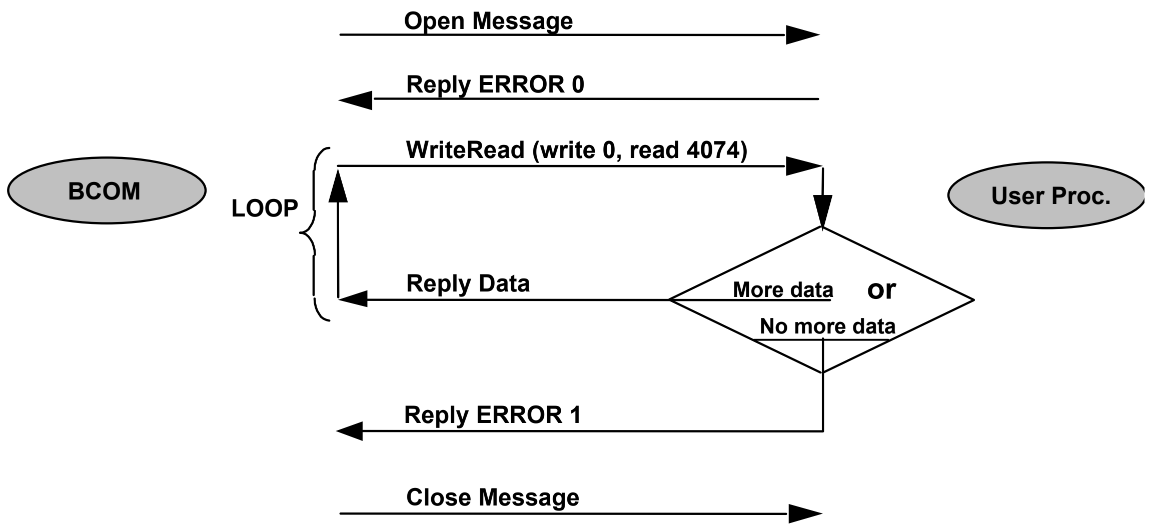
(1) The value of the 2-byte record length field excludes the size of that field;

(2) the size of the application block replied to BCOM can vary in size but must not exceed 4074 bytes (length field of 2 bytes and record of 4072);

(3) the figure of 4072 is Enscribe's maximum record size;

(4) records can be of even or odd value lengths; however, if odd valued, then a pad space is needed before inserting the next record length or end-of-data indicator. Thus, every record length field starts on an even byte boundary.

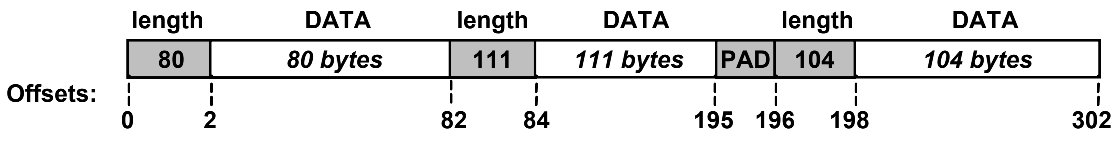
The following diagram illustrates the dialog between BCOM and a User process that sends data to BCOM:



Examples

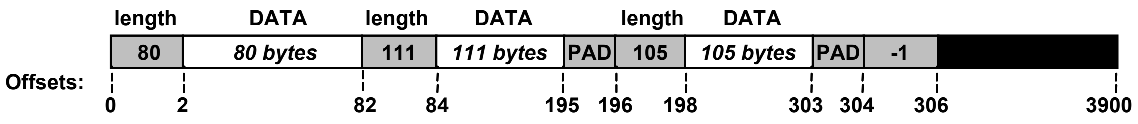
(1) A process sends a 3-record, variable length block:

- record 1 has 80 bytes, record 2 has 111 and record 3, 104;
- the application's reply will be 302 bytes long (301, plus 1 pad space, because of the odd value of one of the record's length) as illustrated below:



(2) A process sends a 3-record, fixed length (3900 bytes) block:

- record 1 has 80 bytes, record 2 has 111 and record 3, 105;
- the application's reply will be 3900 bytes long, inside which the end of data is signalled by a word-aligned end-of-data indicator containing -1. (Note that each odd length data field is completed with a pad space)



Note: *Values in normal-bold characters represent the exact content of the field for each of the above examples, while values in italic-bold indicate the length of the field. The PAD value indicates a 1-character field.*

Using a process to receive blocked data — VAROUT option

The following describes the VAROUT option (which is modeled after the FUP feature of the same name) for receiving blocked records from a remote BCOM location, in a user process.

Request Definition

The request to have a NSK Guardian user process receive blocked data from BCOM can be initiated by a remote BCOM location (e.g. SF) or the local NSK Guardian (e.g. RF).

In either case, to use the VAROUT feature of BCOM, the request definition must meet certain requirements:

- a remote SF request must reference the NSK Guardian user process as a remote file; a local RF must reference it as the local file;
- a remote SF request must include the remote-attribute PARMS=VAROUT; a local RF request must include the local-attribute PARMS=VAROUT

The AUTORESTART and CHECKPOINT request options must NOT be used.

Examples 1

Remote SF (request initiated by a remote platform):

```
SET TYPE            SF
SET LOCAL-LOCATION  %MVSMTL
SET LOCAL-FILE      <MVS.SOURCE.FILE>
SET LOCAL-ATTR01    DISP=SHR
SET REMOTE-LOCATION %TND2
SET REMOTE-FILE     $<USER.PROCESS.NAME>
SET REMOTE-ATTR01   PARAMS=VAROUT
SET ROUTING-CLASS   02
SET AUTO-RESTART    NO
SET CHECKPOINT      0
SET COMPRESSION     NO

ADD #RQSTNO3
```

Examples 2

Local RF (request initiated by the NSK Guardian platform):

```
SET TYPE            RF
SET LOCAL-LOCATION  %TND2
SET LOCAL-FILE      $<USER.PROCESS.NAME>
SET LOCAL-ATTR01    PARAMS=VAROUT
SET REMOTE-LOCATION %MVSMTL
SET REMOTE-FILE     <MVS.SOURCE.FILE>
SET REMOTE-ATTR01   DISP=SHR
SET ROUTING-CLASS   2
SET AUTO-RESTART    NO
SET CHECKPOINT      0
SET COMPRESSION     NO

ADD #RQSTNO4
```

Process Specification

Your process must be running by the time that the local RF BCOM request definition executes, or the remote SF is received and processed by BCOM

BCOM will open your process and write blocks of data to it.

These operations are executed in the following order:

I. Your process must accept BCOM's open message on the $RECEIVE file.

II. Your process must accept the WRITEREAD requests from BCOM: those take the form of a non-zero-length WRITE/zero length READ completion on your $RECEIVE file.

III. Your process will then receive a block of data, built according to the following specifications:

- the IPC buffer has a maximum size of 4074 bytes
- this buffer contains variable length records
- each record begins with a 2-byte length field, followed by the data
- blocks have variable lengths and end with a trailing word-aligned (see below) end-of-data indicator, that is a 2-byte word containing -1 (this does not apply if the block length, without that word, is 4073 or 4074);
- the length field words and the end-of-data word must be aligned on even byte boundaries (a pad space has to be added to the previous record if necessary).

IV. When there is no more data for your process to receive, BCOM will send a close message.

Notes:

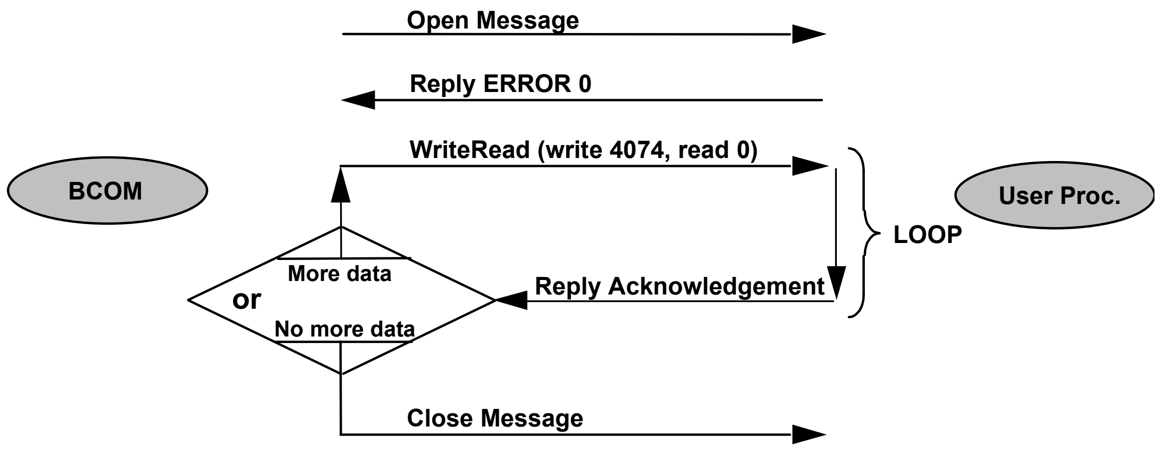
(1) The value of the 2-byte record length field excludes the size of that field;

(2) the size of the application block replied to BCOM can vary in size but will not exceed 4074 bytes (length field of 2 bytes and record of 4072);

(3) the figure of 4072 is Enscribe's maximum record size;

(4) records can be of even or odd value lengths; however, if odd valued, then a pad space will be added before the next record length or end-of-data indicator. Thus, every record length field will start on an even byte boundary.

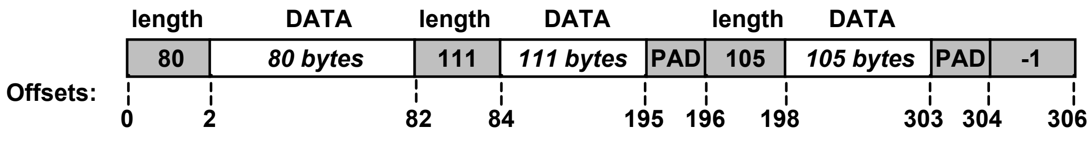
The following diagram illustrates the dialog between BCOM and a User process that receives data from BCOM.



Example

Here is an example of a user process receiving a 3-record block:

- record 1 has 80 bytes, record 2 has 111 and record 3, 105;
- the block will be 306 bytes long (302, plus 2 pad spaces, because of the odd values of two of the record's length, plus 2 for the "-1" end-of-data indicator) as illustrated below:



Note: *Values in normal-bold characters represent the exact content of the field for this example, while values in italic-bold indicate the length of the field. The PAD value indicates a 1-character field.*

Appendix - Command Syntax and Reference Summary

This appendix provides an alphabetical list of the user interface commands that can be used with the NSK Guardian BCOM product.

Included in it are sections that provide:

- A description of individual commands;
- Correct syntax to use when entering commands;
- Special considerations pertinent to each command;
- Examples of how to use each command.

Standard conventions for command syntax are described below.

Syntax Conventions

All of the BCOM manuals follow these conventions to describe commands:

1. BCOM commands appear in Courier type, all in uppercase letters. For example:

    ```
    EXIT
    ```

2. User-selected information that must be inserted in a command appears in Courier type, in lowercase. For example, in the following command:

    ```
    ADD  request-name
    ```

    the user should replace request-name with the name of the request to be added.

3. Optional portions of commands appear within square brackets. For example, in this command:

   ```
   RESET  [ALL]
   ```

   the user may enter either RESET or RESET ALL. And in this command:

   ```
   HELP  [command-name]
   ```

   the user may enter simply HELP, or may optionally add the name of a command.

4. An ellipsis ... is used to indicate that the previous syntax item can be repeated any number of times. For example, in the command:

   ```
   QUEUE  request-name1  [,request-name2,...]
   ```

   the user may enter as many different request-names as desired, as long as at least one is entered.

5. A vertical bar is used to separate a number of different, equally viable alternatives in a command. For example, in the following:

   ```
   HELP  [command-name | ALL]
   ```

   the user may enter either HELP command-name , HELP ALL, but not both.

6. Selectable but required portions of commands may appear within braces. Usually, vertical bars are used to separate a number of different, equally viable alternatives in a command. For example, in the following:

   ```
   RESET { <reset-option> | ALL }
   ```

   the user may enter either RESET ALL or RESET <option>, but not both.

7. Punctuation must be entered exactly as shown

8. Request names begin by the character #.

   Process names begin by the character. $.

   Location names begin by the character %.

Command Security

BCOM offers facilities to control the security on the usage of certain critical BFSINT commands. The Master Operator (MSTOPER) is chosen at installation time. When it is used, the following restrictions apply:

Commands restricted to the MASTER OPERATOR are:

```
ABORT (process)
ACTIVAT STATS
DEACTIVATE STATS
DRAIN
SHUTDOWN
START
SWITCH STATS
WARMSTART
```

Command restricted to users of the MASTER OPERATOR, or the owner of the request:

```
PURGE
```

Commands restricted to users of the MASTER OPERATOR, or the initiator of the queued request:

```
ABORT (request)
QUEUEHOLD
QUEUEDEL
QUEUEREL
```

ABORT

Function:

To force a transfer process to abort, or to stop a request from executing. This command has two main purposes:

1) To ABORT a request or a series of requests.

2) To ABORT a transfer process, or all transfer processes.

Syntax:

The basic syntax for the command is as follows:

```
ABORT  {process-ID | ALL | request-name [, request-name...]}
```

where:

`process-id`	is the name of an active BCOM transfer process to be aborted.
`ALL`	to abort all active transfer processes.
`request-name`	name of the request to be aborted.

Syntactic rules:

1. When aborting a single transfer process, code
```
DRAIN #REQABC
```

2. When aborting more than 1 request, code
```
ABORT #REQ1, #REQ2, #REQ3, ... , #REQn
```

3. When aborting a single active transfer process, code
```
ABORT $BFS62
```

4. When aborting all active transfer processes, code
```
ABORT ALL
```

Consideration:

The ABORT command can be used to interrupt one specific file transfer. When a Download is in progress and the transfer process is aborted, the file which was built with a FUP or SQLCI process, may be left in a corrupted state. If this is the case, delete the data and restart the file transfer.

ABORT should not be used during normal operations, but only in emergency situations. The transfer process names can be obtained through the STATUS ALL command.

Only the Master Operator is able to abort active transfer processes.

Example:

The following ABORT command forces the #C10006 request to abort.

```
~qlist

REQUEST  PRI TP QTIME                STATUS    RLOC     RTC  CHPT  RST  OWNER

#C10006  5   FB 1998-03-31 11:31:13  EXECUTING %Q1D     2    NO    NO   100,255

~abort #C10006
REQUEST #C10006 HAS BEEN ABORTED

~qlist
QUEUE FILE IS EMPTY
```

ACTIVATE STATS

Function:

To activate the Statistics file feature. This command usually issued after a DEACTIVATE STATS command.

Syntax:

```
ACT[IVATE]  STATS
```

Consideration:

1. This command is available ONLY for the Master Operator.
2. BFSMON will create a new file when the next request execution completes.
3. If the last day has been created, a warning message
   ```
   "NO MORE STATISTIC FILE CREATED FOR CURRENT DAY."
   ```
   is issued and Statistics file option remains “DEACTIVATED”
4. If BFSMON was not started with the =BCOM-STATS-FILE DEFINE, the following warning message is issued:
   ```
   "STATISTIC FILE IS NOT DEFINED."
   ```

Example:

```
~ACTIVATE STATS
STATISTIC FILE IS ACTIVATED
```

ADD

Function:

This command is used to add a previously set request to a Request file.

Syntax:

```
ADD request-name
```

Consideration:

Operation will be greatly simplified if significant naming convention is chosen for request identification. Once added, a definition can be removed by a PURGE command.

After the request is successfully added, all working request attributes (as shown in the SHOW command) are reset to blanks, zeroes, or to their default value. SET commands should be used again to set these attributes before another ADD command can be issued.

Example:

The following ADD command will add the new request #EXAMP5 as a copy of $EXAMP4.

```
~LISTREQ
#EXAMP1   #EXAMP2   #EXAMP3   #EXAMP4
~SET LIKE #EXAMP4
~ADD #EXAMP5
REQUEST ADDED
~LISTREQ
#EXAMP1   #EXAMP2   #EXAMP3   #EXAMP4   #EXAMP5
```

ADSM (obsolete – replaced by TSM)

Function:

This command introduces a subcommand for BackHome/TSM option. In BCOM V5.1, this command is replaced by command TSM

Syntax:

```
ADSM subcommand parameter
```

Considerations:

Refer to the *BackHome/TSM Installation and User Guide* for details.

COMMENT

Example:

To support a comment line within a command file.

Syntax:

```
COMMENT [comment-text]
```

where:

`comment-text` is user-defined text.

Considerations:

COMMENT commands can be used to enter comments within a command file. The use of COMMENT is recommended to document OBEY files.

Example 1:

Simple use of the COMMENT command.

```
~COMMENT  ****  THIS IS NOT PROCESSED BY BFSINT  ****
```

Example 2:

COMMENT command used in an OBEY file.

```
~OBEY EXAMPLE
COMMENT ----------------------------------------------------
COMMENT
COMMENT -- OBEY FILE USED TO START THE EXAMPLE PROCESS
COMMENT
COMMENT ----------------------------------------------------
QUEUE #RQ05
~
```

BCOM also supports request comments, which can be entered via the SET COMMENTn command, and are stored on the request file.

DEACTIVATE STATS

Example:

To de-activate the Statistics file feature.

Syntax:

```
DEACT[IVATE]  STATS
```

Considerations:

1. This command is available ONLY for the Master Operator.
2. BFSMON will close the current Statistic file, and mark the option as “INACTIVE”.
3. If BFSMON was not started with the =BCOM-STATS-FILE DEFINE, the following warning message is issued:

```
"STATISTIC FILE IS NOT DEFINED."
```

Example:

```
~DEACTIVATE STATS
STATISTIC FILE IS DEACTIVATED
```

DRAIN

Function:

To stop a transfer process or all transfer processes after completion of transfers that are currently in progress.

Syntax:

```
DRAIN {process-id | ALL}
```

where:

`process-id`	is the name of an active BCOM transfer process.
`ALL`	to drain all active transfer processes.

Syntactic rules:

1. When draining a single transfer process, code
   ```
   DRAIN  $BFS62
   ```
2. When draining all the active transfer processes defined in the local Configuration, code:
   ```
   DRAIN  ALL
   ```

Considerations:

This command is available ONLY for the Master Operator.

The transfer process names can be obtained through the STATUS ALL command.

Example:

The following command drains the $Q1MR process.

```
~status all
*** CURRENT STATUS AT 2001-08-06  13:46:42

  PROCESS             JOBS                MAX    ENABLE    DATA
TYPE    ID      SELECT  STATUS   CONNECTED RQSTS  COMPRESS  SORTED  ENCRYPT
STNDRD  $Q1MR   0       ACTIVE   YES       4      YES       YES     NO
MLTPLX  $Q1MN   0       ACTIVE   NO        1      YES       YES     YES
STNDRD  $Q1MS   0       ACTIVE   YES       3                YES
MLTPLX  $Q1MM   0       ACTIVE   NO        1                YES

~DRAIN $Q1MR
PROCESS $Q1MR DRAINED.

~STATUS $Q1MR
*** CURRENT STATUS AT 2001-08-06  13:47:24
$Q1MR DOES NOT EXIST
```

EXIT

Function:

To terminate the execution of the user interface module.

Syntax:

```
EXIT            OR
<control-Y>
```

Consideration:

When BFSINT encounters the EXIT command while reading a command file (e.g., the IN-FILE), it stops immediately.

On NSK Guardian, the CNTL-Y (control Key + Y) can also be used to terminate execution of the user interface module.

Example:

This command terminates execution of BFSINT, regardless of the mode BFSINT was invoked with.

```
~EXIT
133>
```

FC

Function:

To correct or re-execute a command that has already been executed.

This command is available only on the NSK GUARDIAN BFSINT process.

Syntax:

```
FC   [<command-prefix> | <command-number>]
```

Consideration:

When this command executes, it displays the previous command line and prompts for editing input with a period (.). Please refer to Tandem TACL commands documentation for details on use of this command.

If <command-prefix> is specified, the last command with the same prefix is selected.

<command-number> is the rank of the command since the BFSINT session has started. The rank may be retrieved with the **H[istory]** comand.

HELP

Function:

This command displays help information for a particular command or for all the commands.

Syntax:

```
HELP  [command | ALL]
```

where

command	identifies a BFSINT command
ALL	specifies all BFSINT commands.

Consideration:

If you omit the parameters and simply enter the command itself, the summary command list will be displayed.

Example:

This HELP command displays the syntax for the ADD command.

```
~HELP  ADD
ADD  <#request-name>
```

HISTORY

Function:

This command lists the last commands. Usually, this command is used in conjunction with the FC command.

Syntax:

```
H[ISTORY] [nn]
```

where

nn number of previous commands to be listed. The default is 50.

Consideration:

BFSINT keeps track of the last 50 commands.

This command is similar to the TACL HISTORY command.

Example:

This HISTORY command shows all commands since starting BFSINT.

```
~H

1 LISTLOG
2 START $Q1MR
3 START $Q1MS
4 STATUS ALL
```

NOTE: The ordinal number in front of the command can be used in the FC command.

INFO

Function:

To display file transfer attributes for a particular request or for all the requests. THIS INFORMATION IS TAKEN FROM THE REQUEST FILE.

Syntax:

```
INFO  {request-name | ALL}
```

where

request-name	refers to the file transfer request whose attributes will be displayed
ALL	refers to all file transfer requests.

Example:

The following command displays file transfer attributes for the #DUMP2 request.

```
~INFO  #DUMP2
  REQUEST           = #DUMP
  TYPE              = SEND-FILE
  COMMENTS:
    01 SAMPLE SEND-FILE
  OWNERID           = 130,50
  ROUTING-CLASS     = 2
  PRIORITY-CLASS    = 2
  LOCAL-LOCATION    = %Q1T
  LOCAL-FILE        = $DATA.FILE.TEST
  LOCAL-ATTRIBUTES:
    01 PROCESS-PRIORITY=140
  REMOTE-LOCATION   = %Q1M
  REMOTE-FILE       = MVS.FILE.TEST
  REMOTE-ATTRIBUTES:
    01 DISP=SHR
  AUTO-RESTART      = NO
  CHECKPOINT (REC.) = 0
  COMPRESSION       = NO
  NORMAL-END        = <None>
  ABNORMAL-END      = <None>
  FIELD             = NONE
```

LISTBACKUP

Function:

To display backup executions known in the local backup catalog.

Syntax:

```
LISTBACKUP | LISTBK  remote-object-name [, DETAIL]
```

where

`remote-object-name`	is MVS file name or TSM object name, depending on the server type where the backup was stored. A wildcard (*) can end the remote-object-name.

Considerarion:

This command is now reserved for BackHome/TSM. Please refer to the *BackHome/TSM - Installation and User Guide* for details.

LISTLOG

Function:

To extract and display log messages from the BCOM log file(s).

Syntax:

```
LISTLOG
        [A[ALL]]
        [L[AST] nn]
        [[S[TART] [yyyy-mm-dd] hr:mn:sc] [E[ND] [yyyy-mm-dd] hr:mn:sc]]
```

Consideration:

This command is invalid when EVENT-LOGGING =YES is chosen when logging is started. In this case, please refer to the "*Receiving BCLOM Log messages*" subsection in Appendix B.

If no options are included, all messages for the current day will be displayed.

If ALL (or A) option is specified, all messages in the log files will be displayed.

If LAST (or L) option is specified, list messages in the last "nn" minutes.

If START and/or END are specified, list messages from the specified START date/time to the specified END date/time.

If the Startup PARAM PAUSED-DISPLAY YES was set, LISTLOG pauses at every 20 lines of output.

Examples:

```
LISTLOG LAST 5

LISTLOG START 1996-11-14 10:23:00 END 1996-11-15 09:30:00

LISTLOG S 1996-11-14 10:23:00 E 1996-11-15 09:30:00
```

Notes: *- If using the LISTLOG START … END… command, at least either the START information or the END information must be entered, and when entering the START info and/or the END info, at least either the day or the time info must be entered.*

- For both the START and the END info, the time, if specified, must include the seconds

- For both the START and the END info, if the day if not specified, the current day is assumed

- For the START info, if the time is not specified the beginning of the day is assumed

- For the END info, if the time is not specified the end of the day is assumed

- For ALL, LAST, START and END operands, A, L, S and E abbreviations are allowed.

LISTREQ

Function:

To display current request names residing on the Request file.

Syntax:

```
LISTREQ  [generic-rqst-name]
```

where:

`generic-rqst-name`	displays all file transfer request names matching the generic name entered.

Consideration:

If you omit the parameter of the LISTREQ command, all the file transfer request names will be displayed in alphabetical order.

Example 1:

This command displays all request names.

```
~LISTREQ
#EXAMP10  #EXAMP11  #EXAMP20   #EXAMP21
```

Example 2:

This command displays all the request names beginning with #EXAMP1.

```
~LISTREQ #EXAMP1
#EXAMP10  #EXAMP11
```

Example 3:

The following command displays all the request names beginning with #EX.

```
~LISTREQ  #EX
#EXAMP10  #EXAMP21  #EXAMP52  #EXCHG    #EXTRA
```

LISTSAVEDFILE

Function:

To display the backups of a NSK Guardian file, from the local backup catalog.

Syntax:

```
LISTSAVEDFILE | LISTSF Guardian-file-name [, DETAIL]
```

where

`Guardian-file-name`	is the Guardian file for which backup are looked for. A wildcard (*) can end the Guardian file name.

Considerarion:

This command is now reserved for BackHome/TSM. Please refer to the *BackHome/TSM - Installation and User Guide* for details.

OBEY

Function:

To instruct the BFSINT user interface to execute a series of commands from a source file.

Syntax:

```
OBEY  file-name
```

where:

`file-name`	is the name of the NSK Guardian file from which BFSINT reads commands. This file-name may refer to a disk file. The commands read from the OBEY file must be BFSINT commands, not NSK Guardian TACL commands.

Considerations:

BFSINT reads and executes commands from the named file until it encounters an EXIT command or the end of the file. After the end of the file, BFSINT closes the Obey file and command input reverts to the file from which the OBEY command was read.

Multiple OBEY commands can appear within an Obey file; you can nest Obey files to a depth of four (4).

If BFSINT detects an error while processing a command from an Obey file, it closes this file and returns to its original command stream.

Example:

The following command has BFSINT read and execute the Obey file EXAMPLE.

```
~OBEY  EXAMPLE
COMMENT --------------------------------------------------
COMMENT -- OBEY FILE USED TO START THE BCOM PROCESS
COMMENT --
COMMENT --------------------------------------------------
START $RB01
$RB01 PROCESS STARTED
```

OPEN

Function:

To open the monitor with which the user interface will establish communication.

Syntax:

```
OPEN  monitor-name
```

where:

`monitor-name` is the name of the BCOM monitor process to be opened.

Example:

The following command opens the BMON process $EXAMP, allowing the user to communicate with several BMONs without leaving the BFSINT session.

```
~OPEN  $EXAMP
OPEN COMPLETED, BATCH MONITOR PROCESS NAME=$EXAMP
~
```

The name of the monitor can be in NSK Guardian Expand Network format.

PURGE

Function:

To purge a request from the Request file.

Syntax:

```
PURGE  request-name [request-name, request-name...]
```

where:

`request-name` is the file transfer request to be purged.

Consideration:

You can only use this command to purge requests that are not currently queued.

Example 1:

The following command purges request #EXAMP1.

```
~LISTREQ
#EXAMP1   #EXAMP2   #EXAMP3   #EXAMP4
~PURGE  #EXAMP1
#EXAMP1 PURGED
~
```

Example 2:

The following command purges requests #EXAMP2 and #EXAMP3.

```
~PURGE  #EXAMP2, #EXAMP3
#EXAMP2 PURGED
#EXAMP3 PURGED
~
```

QUEUE

Function:

To queue a file transfer request that will be executed.

Syntax:

```
QUEUE request-name[,request-name][(routing-class,priority-class)]
```

where:

routing-class	is the routing class over which this request will be transferred
priority-class	represents the priority with which this request will be queued

Consideration:

Routing-Class and Priority-Class parameters are optional, but if entered, they will override the request definition. If they are entered, both of them must be entered together (separated by a comma). These parameters apply to all the specified requests. The parameters apply only to this QUEUE not to subsequent queues. (multiple requests may be entered).

If the routing class parameter is entered, this should apply only to the requests being queued.

If any of these requests have normal or abnormal end set, the routing class should not be carried over. But on the other hand the priority must apply on all requests that are internally queued by the normal/abnormal end.

Example 1:

The following command queues request #EXAMP1 for file transfer.

```
~queue #EXAMP1
#EXAMP1 QUEUED, ROUTING CLASS 01, PRIORITY CLASS 1

~qlist

REQUEST  PRI TP QTIME               STATUS    RLOC     RTC  CHPT  RST  OWNER

#EXAMP1  1   FB 1998-10-10 10:15:06 WAITING   %Q1D     1    NO    NO   100,255
```

Example 2:

The following command queues requests #EXAMP2, #EXAMP3 and #EXAMP4.

```
~QUEUE #EXAMP2,EXAMP3,#EXAMP4
#EXAMP2      QUEUED
#EXAMP3      QUEUED
#EXAMP4      QUEUED

~QLIST

REQUEST  PRI TP QTIME                STATUS    RLOC     RTC  CHPT  RST  OWNER

#EXAMP1  1   FB 1998-10-10 10:15:06  WAITING   %Q1D     1    NO    NO   100,255
#EXAMP2  2   FB 1998-10-10 10:21:13  EXECUTING %Q1D2    2    500   NO   100,255
#EXAMP3  5   FB 1998-10-10 10:21:14  SELECT    %Q1D3    5    NO    NO   100,255
#EXAMP4  9   FB 1998-10-10 10:21:14  WAITING   %Q1D     3    NO    NO   100,255
```

QUEUEALT-PRI | QALTP

Function:

Alter the priority-class of a queued request.

Syntax:

```
QUEUEALT-PRI | QALTP  request-name, priority-class
```

where:

request-name	is the file transfer request whose priority-class is to be altered.
priority-class	is the new value for the priority class to be given to the request.

Consideration:

The request to be altered must be in the current queue but not executing.

Example:

The following command changes the priority of the queued request #EXAMP.

```
~QLIST

REQUEST  PRI TP QTIME                STATUS    RLOC     RTC  CHPT  RST  OWNER

#EXAMP   4   FB 1998-01-21 10:15:06  WAITING   %Q1D     2    NO    NO   100,255

~QALTP  #EXAMP,9
#EXAMP PRIORITY HAS BEEN ALTERED

~QLIST

REQUEST  PRI TP QTIME                STATUS    RLOC     RTC  CHPT  RST  OWNER

#EXAMP   9   FB 1998-01-21 10:23:14  WAITING   %Q1D     2    NO    NO   100,255
```

QUEUEDEL | QDEL

Function:

Delete a queued request from the Queue file if the transfer has not yet started.

Syntax:

```
QUEUEDEL | QDEL  request-name [request-name, request-name...]
```

where:

`request-name` is the file transfer request to be deleted.

Consideration:

The request to be deleted must be in the current queue but not executing.

The request will be deleted from the Queue file only.

Example:

The following command deletes the currently queued request #EXAMP1.

```
~QDEL  #EXAMP1
#EXAMP1 DELETED
~
```

QUEUEHOLD | QHOLD

Function:

Hold a queued request from being executed before it has begun. The request will not lose its FIFO order within the queue.

Syntax:

```
QUEUEHOLD | QHOLD  request-name [request-name, request-name...]
```

where:

`request-name` is the file transfer request to be held.

Consideration:

The request(s) to be held must be currently queued, but not executing.

Example 1:

This command holds the queued request #EXAMP1.

```
~QHOLD  #EXAMP1
#EXAMP1 HAS BEEN HELD
~
```

Example 2:

This command tries to hold the queued request #EXAMP2.

```
~QHOLD  #EXAMP2
#EXAMP2  NOT HELD, CURRENTLY EXECUTING
~
```

QUEUELIST

Function:

Display the currently queued requests.

Syntax:

```
QUEUELIST | QLIST
```

Consideration:

Once a file transfer is complete, the request entry is deleted from the queue. The order of the queued requests as they are displayed with this command represents the FIFO order within the queue.

Example :

This command displays all the requests currently queued.

```
~QLIST

REQUEST  PRI TP QTIME               STATUS    RLOC     RTC  CHPT  RST  OWNER

#EXAMP1  1   FB 1998-10-10 10:15:06  HELD      %Q1D     1    NO    NO   100,255
#EXAMP2  2   FB 1998-10-10 10:21:13  EXECUTING %Q1D2    2    500   NO   100,255
#EXAMP3  5   FB 1998-10-10 10:21:14  SELECT    %Q1D3    5    NO    NO   100,255
#EXAMP4  9   FB 1998-10-10 10:21:14  HELD      %Q1D     3    NO    NO   100,255
```

QUEUEREL | QREL

Function:

Release a previously queued request which has been held with the QUEUEHOLD command.

Syntax:

```
QUEUEREL | QREL  request-name [request-name, request-name...]
```

where:

`request-name` is the file transfer request to be released.

Consideration:

This command should only be used in the event that a given request has been held previously with the QUEUEHOLD command; otherwise an error message will be displayed.

Example :

The following command releases the queued request #EXAMP1 from the previously held state.

```
~QLIST

REQUEST  PRI TP QTIME                STATUS    RLOC     RTC  CHPT  RST  OWNER

#EXAMP1  1   FB 1998-10-10 10:15:06  HELD      %Q1D     1    NO    NO   100,255
#EXAMP2  2   FB 1998-10-10 10:21:13  EXECUTING %Q1D2    2    500   NO   100,255
#EXAMP3  5   FB 1998-10-10 10:21:14  SELECT    %Q1D3    5    NO    NO   100,255
#EXAMP4  9   FB 1998-10-10 10:21:14  HELD      %Q1D     3    NO    NO   100,255

~QUEREL #EXAMP1
REQUEST #EXAMP1 HAS BEEN RELEASED

~QLIST

REQUEST  PRI TP QTIME                STATUS    RLOC     RTC  CHPT  RST  OWNER

#EXAMP1  1   FB 1998-10-10 10:15:06  EXECUTING %Q1D     1    NO    NO   100,255
#EXAMP2  2   FB 1998-10-10 10:21:13  EXECUTING %Q1D2    2    500   NO   100,255
#EXAMP3  5   FB 1998-10-10 10:21:14  SELECT    %Q1D3    5    NO    NO   100,255
#EXAMP4  9   FB 1998-10-10 10:21:14  HELD      %Q1D     3    NO    NO   100,255
```

QUEUERESTART | QRESTART

Function:

To release a previously queued request which ended abnormally and is in state HELD for RESTART.

Syntax:

```
QUEUERESTART | QRESTART {<request-name> [,<request-name>]...} [COLD | C]
```

Considerations:

The QUEUERESTART command is used instead of the QUEUE command, because the request is already on the QUEUE file. However, its state indicates that BCOM has terminated its execution due to some errors and now awaits an operators action to either delete or restart this transfer operation.

If the request was defined with the CHECKPOINTING attributes, then a QRESTART command will cause BCOM to re-initiate transfer from the last checkpoint successfully taken.

However, it may be the operator's choice to ignore checkpoints completely and re-initiate the transfer entirely, as if the request was being queued for the first time. In that case, the optional keyword COLD (or C, for short) will tell BCOM to re-initiate the entire transfer from the beginning.

A manual START must be done when AUTO-RESTART = NO has been selected in the SET command.

Example:

This example restarts a previously held request, using the current checkpointing information:

```
~QUEUERESTART  #EXAMP1
#EXAMP1 RESTARTED, ROUTING CLASS 01, PRIORITY CLASS 5, CHECKPOINT 00050
```

QUEUEWAIT | QWAIT

Function:

To initiate a file transfer request and wait for its completion.

Syntax:

```
QUEUEWAIT | QWAIT  request-name [(routing-class, priority-class)]
```

Considerations:

When QUEUEWAIT is issued, the BFSINT session is suspended until the file transfer is completed.

If BFSINT is running in interactive mode, QUEUEWAIT displays a message that the transfer has been initiated. When the request is completed, a message will be displayed and BFSINT will be ready for another command.

When running in batch mode, the status of the request is displayed. If a failure is detected, BFSINT will abend; otherwise BFSINT will continue with the next command.

If BFSINT is run in single command mode, the status of the request will be displayed. If a failure is detected, BFSINT will abend; otherwise, it will terminate normally. The reason for this behavior is to provide feedback to local job schedulers, example : NETBATCH.

The optional parameters include queue override options. These are identical in function to the queue override commands specified in the QUEUE command. As with the QUEUE command, all the override parameters are optional. However, if they are to be used, BOTH of them must be specified (separated by a comma). These parameters are in effect only for this queuing of the request.

Example 1

This command initiates the file transfer request #EXAMP1 and displays the completion status.

```
~queuewait #EXAMP1
REQUEST #EXAMP1 , ROUTING CLASS 1, PRIORITY CLASS 5
#EXAMP1  FILE TRANSFER HAS BEEN INITIATED..
#EXAMP1  HAS COMPLETED SUCCESSFULLY

*** CURRENT STATUS AT 1998-03-31 14:18:24

REQUEST NAME         = #EXAMP1
TYPE                 = FILE-BACKUP
QUEUE TIME           = 1998-03-31 14:18:08
START TIME           = 1998-03-31 14:18:12
END TIME             = 1998-03-31 14:18:24
QUEUE USER ID        = 130,50
RECORD COUNT         = 307
BLOCK COUNT          = 216
BFSBKUP  COMPL. CODE = 0
BFSBKUP  TERM.       = <None>
JOB-ID               = $X4AT
JOB-OBJ              = BFSBKUP
MESSAGE              = TRANSFER SUCCESSFUL
~
```

Example 2

This command initiates the file transfer request #EXAMP2 to be transferred over routing class 2, priority class 5, and displays the completion status.

The request did not complete normally. Since QUEUEWAIT was used, feedback as to the outcome of the transfer is displayed on normal or abnormal completion.

```
~queuewait #EXAMP2 (2, 5)
REQUEST #EXAMP2, ROUTING CLASS 2, PRIORITY CLASS 5
#EXAMP2  FILE TRANSFER HAS BEEN INITIATED..
#EXAMP2  HAS NOT COMPLETED SUCCESSFULLY, JCL ERROR

*** CURRENT STATUS AT 1998-03-31 14:39:32

REQUEST NAME         = #EXAMP2
TYPE                 = FILE-BACKUP
QUEUE TIME           = 1998-03-31 14:39:30
START TIME           = 1998-03-31 14:39:31
END TIME             = 1998-03-31 14:39:32
QUEUE USER ID        = 130,50
RECORD COUNT         = 0
BLOCK COUNT          = 0
         COMPL. CODE = 0
         TERM.       = <None>
JOB-ID               =
JOB-OBJ              =
MESSAGE              = FILE ALLOCATION ERROR
```

RESET

Function:

To re-initialize with spaces or default value all parameters or a specific parameter prior to creating a request.

Syntax:

```
RESET  [reset-option | ALL]
```

where:

reset-option	can be any of the following attributes: ABNORMAL-END AUTO-RESTART CHECKPONT COMMENTnn COMPRESSION FIELD LOCAL-ATTRnn LOCAL-FILE LOCAL-LOCATION NORMAL-END PRIORITY-CLASS REMOTE-ATTRnn REMOTE-FILE REMOTE-LOCATION ROUTING-CLASS TYPE
ALL	reset all attributes:

Considerations:

If RESET is entered with no parameters, BFSINT resets all attributes to spaces or to default value. The following attributes have default value:

LOCAL-LOCATION	Local Location from Configuration.
PRIORITY-CLASS	Depending on the value set by the Startup PARAM RQST-PRIORITY-CLASS, or by the SET RQSTDEFAULT command. If none is set, 5 is used.
ROUTING-CLASS	depending on the value set by the Startup PARAM RQST-ROUTING-CLASS or by the SET RQSTDEFAULT command.
RLOC	Depending on the value set by the Startup PARAM RQST-RLOC or by the SET RQSTDEFAULT command.

Example 1:

In this example, parameters are set after those of the #EXAMP1 request and then the RESET command resets the value of the COMPRESSION parameter to NO (default value).

```
~SET LIKE  #EXAMP1
~SHOW

REQUEST PARAMETERS
  REQUEST-TYPE        = SEND-FILE
  COMMENTS:
    01 This is an example for SEND-FILE request to
    02 a currently existing MSV file.
  ROUTING-CLASS         = 02
  PRIORITY-CLASS        = 2
  LOCAL-LOCATION        = %TAND1
  LOCAL-FILE            = #DATA01.FILE.TEST
  LOCAL-FILE-ATTRIBUTES:
    01    PROCESS-PRIORITY = 170
    02    REPORT OFF
  REMOTE-LOCATION       = %MVS1
  REMOTE-FILE           = MVS.TARGET.DSN
  REMOTE-FILE-ATTRIBUTES:
    01    DISP= SHR
  AUTO-RESTART          = NO
  CHECKPOINT            = NO
  COMPRESSION           = YES
  ABNORMAL-END          = <None>
  NORMAL-END            = <None>

~RESET COMPRESSION

~SHOW

REQUEST PARAMETERS
  REQUEST-TYPE        = SEND-FILE
  COMMENTS:
    01 This is an example for SEND-FILE request to
    02 a currently existing MSV file.
  ROUTING-CLASS       = 2
  PRIORITY-CLASS      = 2
  LOCAL-LOCATION      = %TAND1
  LOCAL-FILE          = #DATA01.FILE.TEST
  LOCAL-FILE-ATTRIBUTES:
    01    PROCESS-PRIORITY = 170
    02    REPORT = OFF
  REMOTE-LOCATION     = %MVS1
  REMOTE-FILE         = MVS.TARGET.DSN
  REMOTE-FILE-ATTRIBUTES:
    01    DISP= SHR
  AUTO-RESTART        = NO
  CHECKPOINT          = NO
COMPRESSION           = NO
ABNORMAL-END          = <None>
NORMAL-END            = <None>
```

Example 2:

In this example, parameters are set after those of the #EXAMP2 request and then the RESET command resets the value of the PRIORITY-CLASS to 5 (default value).

```
~SET LIKE  #EXAMP2
~SHOW

REQUEST PARAMETERS
  REQUEST-TYPE        = SEND-FILE
  COMMENTS:
    01 This is an example for SEND-FILE request to
    02 a currently existing MSV file.
  ROUTING-CLASS       = 2
  PRIORITY-CLASS      = 2
  LOCAL-LOCATION      = %TAND1
  LOCAL-FILE          = #DATA01.FILE.TEST
  LOCAL-FILE-ATTRIBUTES:
    01    PROCESS-PRIORITY = 170
    02    REPORT = OFF
  REMOTE-LOCATION     = %MVS1
  REMOTE-FILE         = MVS.TARGET.DSN
  REMOTE-FILE-ATTRIBUTES:
    01    DISP=SHR
  AUTO-RESTART        = NO
  CHECKPOINT          = NO
  COMPRESSION         = NO
  ABNORMAL-END        = <None>
  NORMAL-END          = <None>

~RESET PRIORITY-CLASS

~SHOW

REQUEST PARAMETERS
  REQUEST-TYPE        = SEND-FILE
  COMMENTS:
    01 This is an example for SEND-FILE request to
    02 a currently existing MSV file.
  ROUTING-CLASS       = 2
  PRIORITY-CLASS      = 5
  LOCAL-LOCATION      = %TAND1
  LOCAL-FILE          = #DATA01.FILE.TEST
  LOCAL-FILE-ATTRIBUTES:
    01    PROCESS-PRIORITY = 170
    02    REPORT = OFF
  REMOTE-LOCATION     = %MVS1
  REMOTE-FILE         = MVS.TARGET.DSN
  REMOTE-FILE-ATTRIBUTES:
    01    DISP=SHR
  AUTO-RESTART        = NO
  CHECKPOINT          = NO
  COMPRESSION         = NO
  ABNORMAL-END        = <None>
  NORMAL-END          = <None>
```

Example 3:

The following RESET command will provide the user with a display of parameter values that are either an empty field or a default value. Using the SET commands described in the next section, the user can define a request and add it to the BCOM requests database with the ADD command.

```
~SET LIKE  #EXAMP3
~SHOW

REQUEST PARAMETERS
  REQUEST-TYPE        = SEND-FILE
  COMMENTS:
    01 This is an example for SEND-FILE request to
    02 a currently existing MSV file.
  ROUTING-CLASS       = 4
  PRIORITY-CLASS      = 2
  LOCAL-LOCATION      = %TAND1
  LOCAL-FILE          = $VOL.SUBVOL.FILE
  LOCAL-FILE-ATTRIBUTES:
    01    PROCESS-PRIORITY = 150
  REMOTE-LOCATION     = %MVS1
  REMOTE-FILE         = MVS.TARGET.DSN
  REMOTE-FILE-ATTRIBUTES:
    01    DISP=SHR
  COMPRESSION         = NO
  ABNORMAL-END        = <None>
  NORMAL-END          = <None>

~RESET (or RESET ALL)

~SHOW

REQUEST PARAMETERS
  REQUEST-TYPE        = <None>
  ROUTING-CLASS       = NONE
  PRIORITY-CLASS      = 5
  LOCAL-LOCATION      = %TAND1
  LOCAL-FILE          = <None>
  REMOTE-LOCATION     = <None>
  REMOTE-FILE         = <None>
  NORMAL-END          = <None>
  ABNORMAL-END        = <None>
```

SET

Function:

To set request attributes individually or, using the LIKE parameter, to set attributes to match those of an existing request.

Syntax:

```
SET  {keyword | LIKE request-name}
```

Where

keyword belongs to the following categories:

I. the type keyword, which determines what kind of file transfer the request is; for example, a job submission, a backup or restore, etc.

```
TYPE                  *
```

II. keywords related to the local platform:

```
LOCAL-LOCATION        *
LOCAL-FILE
LOCAL-ATTRIBUTES
```

III. keywords related to the remote platform:

```
REMOTE-LOCATION       *
REMOTE-FILE
REMOTE-ATTRIBUTES
```

IV. keywords related to other aspects of the request:

```
ABNORMAL-END
AUTO-RESTART
CHECKPOINT
COMMENT
COMPRESSION
FIELD
NORMAL-END
PRIORITY-CLASS        *
ROUTING-CLASS         *
```

*Minimum required

A table at the end of this section indicates when the parameters of the II, III and IV categories are mandatory or optional, depending on the type of request they are used with.

I. The TYPE Keyword

```
TYPE file-transfer-type
```

where **`file-transfer-type`** is one of the following:

FB (FILE-BACKUP) specifies a backup of the source files defined in the LOCAL-FILE to the destination files specified in the REMOTE-FILE on the system defined in the REMOTE-LOCATION.

FR (FILE-RESTORE) specifies a restore of the files from the REMOTE-FILE to the LOCAL-FILE.

SF (SEND-FILE) specifies a file transfer from the local location to the remote location.

RF (RECEIVE-FILE) specifies a file transfer from the remote location to the local location.

LJ (LOCAL-JOB) specifies that this job submission request will execute on the local node. The keyword used to specify the input file (LOCAL-FILE, REMOTE-FILE) determines whether the job data is local to this platform or resides on a remote node.

RJ (REMOTE-JOB) specifies that this Job submission request will execute on a remote-location node. The keyword used to specify the input file (LOCAL-FILE, REMOTE-FILE) determines whether the Job data is local to this platform or resides on a remote node.

RQ (REMOTE-FILE) specifies that this is a request for BCOM to queue a request definition at a remote location. REMOTE-FILE indicates the file name on the REMOTE-LOCATION node where the file participating in this transfer resides. Also used to specify the name of a remote request to be queued at a remote location.

II. Keywords related to the LOCAL PLATFORM

`LOCAL-LOCATION location-name`

Specify the logical name of the source location, or one BCOM location name of the BFSINT program that processes one command. The name must exist in the Configuration file. Location names begin with the % sign.

`LOCAL-FILE <File name>  or`
`LOCAL-FILE DNS-ALIAS=<Alias name>`

Alias name is an Alias defined in the DNS system. Usage of DNS requires that the DNS Manager be specified at startup of the BFSMON Monitor. DNS aliases are supported only by SNA APPC transfer.

`LOCAL-ATTRnn identifier=value`

is used to define specific attributes for the LOCAL file and location. Various attributes may be entered, therefore the syntax allows for an identifier followed by = and then the value. The NSK Guardian identifier may be one of the following:

`BRPARAMS=NSK Guardian Backup/Restore options`

For request types FB/FR, specifies the fileset and options of NSK Guardian Backup/Restore utility Refer to NSK Guardian manual for valid options.

For backup and restore requests, the backup or restore command can be retrieved from an user edit file, by specifying BRPARAMS=INFILE file-name. The edit file must contains the whole Backup command, using =BCOM-TAPE as tape device name.

`CATALOG-RETENTIONDAYS=number-of-days`

For request types FB/FR creating a non-GDG MVS file, set an expiry date in the created backup catalog entries.

The default value is the value specified by the JCL remote attribute EXPDT or RETPD. CATALOG-RETENTIONDAYS overrides any value specified in EXPDT or RETPD, and must be specified if EXPDT and RETPD are not.

`CATALOG-TIME=yyyy-mm-dd hr:mn:sc`

For request types FR in BackHome/TSM, chooses a version of a backup according to the local timestamp, as displayed by the LISTBK command.

`JOB-WAITED=yes|no    (default is NO)`

Indicate whether a job submission request should wait for the termination of the job before completing.

`NOCATALOG`

Used when BCOM is configured for using a local backup catalog, and the request must execute as if there is no backup catalog.

`NORMAL-IO-COMPLETION-CODES x,y,z | ALL`

Specify the list of IO-process completion codes that are acceptable for a request to be successful. This optional attributes overrides the configuration value set by the same keyword.

`OUTFILE=NSK Guardian-file-name`

Specify the name of the outfile to be used by TACL for LJ (Local Job), FB (File Backup) or FR (File Restore) type requests.

`PARAMS=process-specific-parameter`

Allow you to pass parameters to the process or job being created by BCOM. For example, to pass data management attributes to FUP, SQLCI.

Examples:

SORTED	(always included when transferring to NSK Guardian)
APPEND	(adding to entry-sequenced for Relative record files)
SHARE	(only valid option for Send-File request type).
PROTECTED	(the default, as opposed to SHARE).
VARIN | VAROUT	(For sending data to receiving it from a process).

`PROCESS-PRIORITY=value`

Allows assigning a Guardian process priority to the I/O process, backup/restore, utility, or NSK Guardian TACL jobs referred by the request

`REPORT=ON | OFF`

Specify ON if your interpretation is to have a print control character; OFF for no print control character interpretation. OFF is the default.

III. Keywords related to the REMOTE PLATFORM

`REMOTE-LOCATION %<location-name>`

where location-name refers to the logical name of the remote location, which must exist in the Configuration file. Begins with a % sign.

`REMOTE-FILE file-name`

specify the name of the remote file.

`REMOTE-ATTRnn identifier=value`

where various sets of identifiers can be used, depending on the platform type.

<u>For an MVS node:</u>

```
DCB=(DSN,LRECL=xxxx,BLKSIZE=xxxx,DEN=n,DSORG=aa,RECFM=xxxx)
```

Where possible values are:

DSN	reference to a dataset name for its attributes	
LRECL	5 car. max. (numeric)	
BLKSIZE	5 car. max. (numeric)	
DEN	2	(800 bpi)
	3	(1600 bpi)
	4	(6250 bpi)
DSORG	PS	(Physical Sequential)
	PO	(Partitioned Organization)
RECFM	F	(fixed)
	V	(variable)
	FB	(fixed, blocked)
	VB	(variable, blocked)
	U	(unstructured)
	VS	(variable, spanned)

```
DISP=(Status, Normal-Disp, Abnormal-Disp)
```

Specify the disposition of the MVS data set . Possible values are:

Status	`NEW`
	`MOD`
	`OLD`
	`SHR`
	`SCR` (for "Scratch")
	(Note: The SCR value is BCOM-specific; it creates a new dataset if none exists or deletes the existing dataset and creates a new one.)
Normal-Disp	`KEEP`
	`DELETE`
	`CATLG`
	`UNCATLG`
Abnormal-Disp	`KEEP`
	`DELETE`
	`CATLG`
	`UNCATLG`

Any of these three parameters can be overwritten, as defaults are defined for each one.

```
LABEL=(MVS-tape-label-info)
```

Specify information pertaining to a tape on the MVS. Used when creating tape datasets on the MVS, or when referring to a specific volume sequence number of an uncataloged MVS tape acting as the FROM-FILE.

The format of this attribute is:

```
   LABEL=(NN,TYPE,Password Protection,IN | OUT,
          RETPDnnnn | EXPDT=yyddd or yyyy/ddd)
```

Possible values are:

NN	2 numeric bytes and is the sequence number
TYPE	`SL`
	`NL`
	`BLP`
	`NSL`
Password Protection	`NOPWREAD` (Read only on the DSN)
	`PASSWORD` (password needed to open and delete the DSN)
	(Note: for no protection, do not put anything)
RETPD	retention period in days (numeric)
EXPDT	expiration date

```
SPACE=(TYPE,PRIM)          OR
SPACE=(TYPE,(PRIM,SECOND,DIR),RLSE)
```

Specify space information for the MVS data set. Valid only when creating a disk dataset on the MVS. Possible values are:

TYPE	CYL TRK (Note: allocation of space in cylinder or track units; the average record length cannot exceed 65,535 bytes)
PRIM	number of units to allocate
SECOND	number of units to allocate if the primary allocation is exceeded
DIR	reserves space for the member names of partitioned data sets
RLSE	to release all unused space when a data set is closed, after having been opened for output and written into

```
UNIT=([xxxx][,number][,DEFER])
```

Specify the UNIT upon which the MVS data set resides. When creating a tape on the MVS, UNIT should specify the TAPE identifier. If you wish to control the placement of new datasets on an MVS, UNIT should be specified. If MVS datasets are catalogued, UNIT need not be specified. Possible values are:

xxxx	hardware address device type group name
number	the number of volumes to be mounted on (1 to 59) in parallel
DEFER	indicates to delay the allocation of the resource at opening of the file.

```
VOL=(MVS-dataset-volume-information)
```

Specify the volume upon which the MVS dataset resides. Users may optionally specify the maximum volume-count that a transfer can use, by entering a two-digit number instead of the volume list.

The format of this attribute is:

`(a) VOLUME LIST`	The format of this attribute for a volume list is: `VOL=(AAAAAA,BBBBBB,CCCCCC)` with up to 3 volume names of up to 6 characters each
`(b) VOLUME COUNT`	The format of this attribute for setting the maximum number of volumes is: `VOL=mm` where mm defaults to 5 if not specified.

```
SUBSYS=(SUBSYS Name)       or
SUBSYS=(SUBSYS Name>, <PARM1>,<PARM2>,<PARM3>)
```

Specify 4 bytes (SUBSYSTEM)   OR

Specify 4 bytes (SUBSYSTEM) with up to 3 PARMS, with 15 bytes max. for each.

`DATACLAS=`(Data Class)	Specify SMS Data Class.
`MGMTCLAS=`(Management Class)	Specify SMS Management Class.
`STORCLAS=`(Management Class)	Specify SMS Storage Class.

For BackHome/TSM specific values:

Please refer to *BackHome/TSM Installation and User Guide* for details.

For a Windows NT node:

I. For File Transfers

`TYPE=Data Structure`

This parameter can be valued UNSTRUCTURED (default), TEXT or RECORD.

`RECFORMAT=Record Format`

This parameter can be valued FIXED (default) or VARIABLE.

`OFLAG=Writing Method`

This parameter can be valued SUPERSEDE (create or replace — default) APPEND, CREATE (if it does not exist) or TRUNCATE (replace).

`RECLEN=Record Length`

This parameter must be valued between 1 to 32767 (default).

II. For Local Job Submission

`JOB-WAITED = yes/no`

Indicate whether a job submission request should wait for the termination of the job before completing. Default = NO

`PARMS=Parameters`

The parameters the job needs.

IV. Keywords related to OTHER ASPECTS of the request

```
ABNORMAL-END request-name
```

where *request-name* is the name of request to be submitted if the current request terminates unsuccessfully.

```
AUTO-RESTART yes|no     (default is NO)
```

instruct BCOM to automatically start a transfer from the last checkpoint value. The maximum number of times that the restart will be attempted is defined in the Configuration file. AUTO-RESTART will only be attempted if the reason for the failure does not preclude a retry. Failures that may be retried are usually transmission error or session error related. In these circumstances, BCOM will re-attempt the transfer.

```
CHECKPOINT record count   (50-999)    (SF and RF requests)
CHECKPOINT number-of-meg  (250-9999)  (FB requests)
```

where *record count* or *number-of-meg* is the interval after which a checkpoint will be taken.

```
COMPRESSION yes|no    (default is NO)
```

where *yes* specifies that compression should be used for this file transfer.
Yes authorized if also set to yes in the configuration file

```
COMMENT comment-line
```

To provide comment lines within a request definition. A Maximum of 5 contiguous comment lines is supported.

```
FIELD offset,length| ALL
```

where the parameters specify locations within a record for which no translation will be performed. A warning will be issued at file transfer time if an offset,length is specified which exceeds the record size of the file.

Notes:

(1) Multiple SET FIELD statements can be entered as they are cumulative. Thus entry of a second SET FIELD is added to the previous values rather than eliminating those values.

(2) Data translation is not required for backup/restore (i.e. you can set FIELD ALL in the request definition). However, be sure that whatever value has been specified in the backup request definition is the SAME specified when defining corresponding restore request.

```
NORMAL-END request-name
```

include the name of the request to be submitted if the current request terminates successfully.

```
PRIORITY-CLASS value
```

where *value* refers to the priority value of the queued request, which may be a number from 0 to 9. If no priority value is set, the default priority is 5.

```
ROUTING-CLASS class
```

where *class* refers to the number of the routing class to be used to transfer the queued request. Character length 2.

REQUEST PARAMETERS	**MANDATORY PARAMS for REQUEST TYPE**							**DEFAULT VALUE (comment)**
	FILE TRANSFER		**Back Rest**	**LOCAL JOB (LJ)**		**REMOTE JOB (RJ)**		
	SF	**RF**	**FB FR**	**local data**	**rmt data**	**local data**	**rmt data**	
AUTO-RESTART								NO
CHECKPOINT				inv	inv	inv	inv	0 (NO CHKP)
COMPRESSION				inv			inv	NO
FIELD				inv			inv	TRANSLATE
LOCAL-LOCATION	√	√	√	√	√	√	√	Note 3
LOCAL-ATTRnn			Notes 1,4	Note 1	Note 1		inv	
NORMAL-END								
ABNORMAL-END								
PRIORITY-CLASS	√	√	√	√	√	√	√	Note 5
ROUTING-CLASS	√	√	√		√	√	√	
TYPE	√	√	√	√	√	√	√	
LOCAL-FILE	√	√	inv	√	inv	√	inv	
REMOTE-LOCATION	√	√	√	inv	√	√	√	Note 3
REMOTE-FILE	√	√	√	inv	√	inv	√	
REMOTE-ATTRnn	Note 2	Note 2	Note 2	inv	Note 2	inv	Note 2	

inv = parameter is invalid

√ = available

NOTES:

1 using the OUTFILE = identifier specify the Tandem file name which will be used as the output file for the TACL/BACKUP/RESTORE

2 see FILEATTR parameter in the BCOM for Windows NT User Guides; the File attributes for Tandem in the BCOM for NSK Guardian User Guides; the JCL attributes in the *BCOM for MVS – User's Guide*.

3 defaults to the location of the BCOM monitor which is opened by the current BFSINT process

4 using the BRPARAMS = identifier, specify the files-set and options for the BACKUP/RESTORE request, multiple ATTRnn may be used to continue the fileset and options.

5 Defaults to 5.

Example 1:

The following commands are used to set up a request, then add it to the Request file.

```
~SET TYPE          SF
~SET LOCAL-LOC     %TAND1
~SET LOCAL-FILE    $VOL.SUBVOL.FILE
~SET LOCAL-ATTR01  PROCESS-PRIORITY = 50
~SET REMOTE-LOC    %MVSMTL
~SET REMOTE-FILE   &&TEMP
~SET REMOTE-ATTR01 UNIT=SYSDA
~SET REMOTE-ATTR02 DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
~SET REMOTE-ATTR03 DISP=(NEW,DELETE)
~SET REMOTE-ATTR04 SPACE=(TRK,(5,5))
~SET ROUTING-CLASS 3

~ADD EXAMP1
REQUEST ADDED
```

Example 2:

With the following SET LIKE command, all the current attributes will be set to the values of those in the #EXAMP1 request.

```
~set like #EXAMP1
~show

REQUEST PARAMETERS
  REQUEST-TYPE        = SEND-FILE
  COMMENTS:
    01 This is an example for SEND-FILE request.
  ROUTING-CLASS       = 3
  PRIORITY-CLASS      = 2
  LOCAL-LOCATION      = %TAND1
  LOCAL-FILE          = $VOL.SUBVOL.FILE
  LOCAL-FILE-ATTRIBUTES:
    01    PROCESS-PRIORITY = 150
  REMOTE-LOCATION     = %MVSMTL
  REMOTE-FILE         = $TEMP
  REMOTE-FILE-ATTRIBUTES:
    01    DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
    02    DISP=(NEW, DELETE)
    03    SPACE=(TRK,(5,5))
  COMPRESSION         = NO
  ABNORMAL-END        = <None>
  NORMAL-END          = <None>
  FIELD(S)   = OFF,LEN      OFF,LEN      OFF,LEN
                 4,2          10,4
```

Example 3:

The following is an example of a typical DISP command.

```
SET REMOTE-ATTR1 DISP=(NEW,CATLG,DELETE)
SET REMOTE-ATTR2 SPACE=(CYL,(10,2))
SET REMOTE-ATTR3 DCB=(RECFM=VB.LRECL=200,BLKSIZE=204)
```

This would be used when creating a new dataset on the MVS. This newly created dataset will be catalogued at transfer termination. Notice that when going to DASD with a DISP of NEW, both SPACE and DCB are usually required.

Example 4:

The following is an example for defining a TAPE.

```
SET REMOTE-ATTR1 LABEL=(2,SL)
SET REMOTE-ATTR2 VOL=D35P12
SET REMOTE-ATTR3 UNIT=TAPE
```

Example 5:

An example of the DCB command for a file transfer to an MVS generation data group (GDG).

```
SET REMOTE-ATTR1 DCB=(MODEL.JCL,BLKSIZE=32000)
```

SET RQSTDEFAULT

Function:

To set Request attributes default.

Syntax:

```
SET  RQSTDEFAULT, <attr-name> <attr-value>
```

where:

`<attr-name>`	is name of a request attribute. As followed:
	`RQST-RLOC`
	`RQST-PRIORITY-CLASS`
	`RQST-ROUTING-CLASS`

`<attr-value>`	is a value legal to the specified attribute.

Example:

Set the default value for the Remote-Location attribute.

```
SET RQSTDEFAULT, RQST-RLOC %Q1D
```

SHOW

Function:

To display the current request attributes, previously set by the SET command.

Syntax:

```
SHOW
```

Example 1:

The SHOW command displays the current file transfer attributes.

```
~show

REQUEST PARAMETERS
  REQUEST-TYPE        = SEND-FILE
  COMMENTS:
    01 SAMPLE SEND-FILE REQUEST AND AN EXISTING MSV FILE
  ROUTING-CLASS       = 2
  PRIORITY-CLASS      = 2
  LOCAL-LOCATION      = %Q1T
  LOCAL-FILE          = $DATA.FILE.TEST
  LOCAL-FILE-ATTRIBUTES:
    01    PROCESS-PRIORITY=140
  REMOTE-LOCATION     = %Q1M
  REMOTE-FILE         = MVS.FILE.TEST
  REMOTE-FILE-ATTRIBUTES:
    01    DISP=SHR
  COMPRESSION         = YES
  NORMAL-END          = <None>
  ABNORMAL-END        = <None>
  FIELD(S)   = OFF,LEN      OFF,LEN      OFF,LEN
                   4,2         10,4
```

SHOW RQSTDEFAULT

Function:

To display the current request attributes default, previously set by the SET RQSTDEFAULT or from the Start-up PARAM.

Syntax:

```
SHOW RQSTDEFAULT
```

Example:

The SHOW RQSTDEFAULT command displays the current default request attributes.

```
~SHOW RQSTDEFAULT
REMOTE-LOCATION     = %Q1D
PRIORITY-CLASS      = 5
ROUTING-CLASS       = 1
```

SHUTDOWN

Function:

To shutdown the BCOM environment in a coherent fashion.

Syntax:

```
SHUTDOWN  or
SHUTDOWN!
```

Considerations:

This command is available ONLY for the Master Operator.

When the SHUTDOWN command is executed, shutdown will only occur after all current file transfers are completed and all connected BFSINT processes terminate. Subsequently, all transfer modules are stopped.

When the SHUTDOWN command with the exclamation point is executed (i.e. SHUTDOWN!), shutdown will occur after all the current file transfers are completed. It will ignore all connected BFSINT processes. It will bring down the environment regardless of the BFSINT users.

Example:

This SHUTDOWN command shuts down BCOM in the environment from which the command was executed.

```
~SHUTDOWN
BMON PROCESS SHUTTING DOWN
~
```

This SHUTDOWN command shuts down BCOM in the environment from which the command was executed and will wait for the current file transfers to terminate. Once those are complete it will bring down the environment regardless of any current BFSINT sessions.

```
~SHUTDOWN!
BMON PROCESS SHUTTING DOWN HARD
~
```

START

Function:

The START command is used to start one or many transfer processes that have been configured in the Configuration file. If the keyword "AUTO-STARTUP NO' is specified in the configuration file, then all transfer processes must be started explicitly using the START command. Transfer processes ABORTed or DRAINed must subsequently be STARTed to make them available to BCOM for request processing.

Syntax:

```
START {process-id | ALL}
```

where:

process-id	is the name of the BCOM transfer process configured under an appropriate routing-class, which in turn is configured under an appropriate remote-location.
ALL	Start all inactive transfer processes defined to that BMON.

Considerations:

This command is available ONLY for the Master Operator.

As stated above, all entities referred to in the START command must be defined as a valid entity occurrence in the BCOM Configuration file. See the BCOM Installation & Operations Guide on NSK Guardian for details on configuring the environment to BCOM on the NSK Guardian.

Please note that the START command must be processed through the BFSINT interface, in any of the modes described in "Section 1 — User Interface Descriptions".

Example 1:

This Obey file will start a BFS62 process $EXAMP on CPU 1. The priority is set to 170, and 3 requests are allowed at the same time.

```
~OBEY  BCOMFSV.BINTST1
COMMENT  -------------------------------------------------
COMMENT  --  OBEY USED TO START THE TRANSFER PROCESS
COMMENT  --  TITLED $BFS62 IN THE CONFIGURATION FILE
COMMENT  -------------------------------------------------
START $BFS62
```

Example 2:

This START command will start all configured but inactive transfer processes for all configured remote locations.

```
~ START ALL
PROCESS $BFRQ1  STARTED
PROCESS $BFRQ2  STARTED
PROCESS $BFSR1  STARTED
PROCESS $BFSR2  STARTED
```

STATUS

Function:

As the command name implies, the STATUS command displays the STATUS of various components in the BCOM environment. The STATUS command has two main functions, i.e., displaying-playing the status of a request-name, and displaying the status of a transfer process.

If no operands are supplied, BCOM will display the status of the BCOM environment in general. This includes all the program-object-files defined for the makeup of the environment as well as a display of all the transfer processes currently running under the remote locations and routing-classes of this local BCOM.

Syntax:

```
STATUS [process-id]
       [remote-location]
       [request-name[,request-name2]]
       [ALL]
```

where:

process-id	is the name of the transfer process whose status output is desired. Transfer process name begins with the $ sign.
Request-name	is the file transfer request whose status is to be displayed. Request names begins with # sign.
Remote-location	to display status of all active requester processes associated with the given remote-location. Remote-location name begins with the % sign.
ALL	to display status of all active transfer processes.

If no parameter is specified, the Status of the entire ETI-NET File Server environment is displayed, including the Local-location.

Example 1:

The STATUS ALL command displays the status of all active transfer processes.

```
~status all
*** CURRENT STATUS AT 2001-08-06  13:46:42

  PROCESS             JOBS                    MAX    ENABLE    DATA
TYPE    ID      SELECT  STATUS   CONNECTED RQSTS  COMPRESS  SORTED  ENCRYPT
STNDRD  $Q1MR   0       ACTIVE   YES       4      YES       YES     NO
MLTPLX  $Q1MN   0       ACTIVE   NO        1      YES       YES     YES
STNDRD  $Q1MS   0       ACTIVE   YES       3                YES
MLTPLX  $Q1MM   0       ACTIVE   NO        1                YES
```

Example 2 - Status of Requester Standard Process

The following STATUS command displays the status of the $Q1MR requester standard transfer process.

```
~STATUS $Q1MR
*** CURRENT STATUS AT 2001-08-06  13:47:16

  PROCESS            JOBS                     MAX    ENABLE    DATA
TYPE    ID      SELECT  STATUS   CONNECTED RQSTS  COMPRESS  SORTED  ENCRYPT
STNDRD  $Q1MR   0       ACTIVE   NO        4      YES       YES     NO

ROUTING

 REMOTE    REMOTE            ROUTING
 LOCATION  APPLID            CLASS

 %Q1M      Q1M               1 2 3 4 11 12 13 14 15

 TPN
 Q1M

 ENCRYPTION: NO

 LOGICAL UNIT                REQUEST  NB BLOCKS  STATUS   ERROR  ORIGIN
 NAME                        NAME     TRANSFERED

 \ETINET.$Q1R1.#Q1RMU09      NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU10      NONE                OUTSTND  0      NONE
```

Example 3- - Status of Server Standard Process

The following STATUS command displays the status of the $Q1MS server standard transfer process.

```
~STATUS $Q1MS
*** CURRENT STATUS AT 2001-08-06  14:01:14

  PROCESS            JOBS                  MAX    ENABLE    DATA
TYPE    ID      SELECT  STATUS   CONNECTED RQSTS  COMPRESS  SORTED
STNDRD  $Q1MS   0       ACTIVE   YES        4                YES


 LOGICAL UNIT                 REQUEST  NB BLOCKS  STATUS   ERROR  ORIGIN
 NAME                         NAME     TRANSFERED

 \ETINET.$Q1R1.#Q1RMU01       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU02       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU03       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU04       NONE                OUTSTND  0      NONE
```

Example 4 - Status of Requester Multiplex Process

The following STATUS command displays the status of the $Q1MM requester transfer process. (Note that a multiplex requester can have only one routing group)

```
~STATUS $Q1MM
*** CURRENT STATUS AT 2001-08-06  14:12:31

  PROCESS            JOBS                  MAX    ENABLE    DATA
TYPE    ID      SELECT  STATUS   CONNECTED RQSTS  COMPRESS  SORTED  ENCRYPT
MLTPLX  $Q1MM   0       ACTIVE   YES       1      YES       YES     YES

ROUTING

 REMOTE    REMOTE             ROUTING
 LOCATION  APPLID             CLASS

 %PHM      PHM                2 3 4 5 6

 TPN
 PHM

 ENCRYPTION: YES

 LOGICAL UNIT                 REQUEST  NB BLOCKS  STATUS   ERROR  ORIGIN
 NAME                         NAME     TRANSFERED

 \ETINET.$Q1R1.#Q1RMU15       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU16       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU17       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU18       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU19       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU20       NONE                OUTSTND  0      NONE
```

Example 5 - Status of Server Multiplex Process

The following STATUS command displays the status of the $Q1MN server transfer process.

```
~STATUS $Q1MN
*** CURRENT STATUS AT 2001-08-06  14:16:56

  PROCESS            JOBS                  MAX    ENABLE    DATA
TYPE    ID      SELECT  STATUS   CONNECTED RQSTS  COMPRESS  SORTED
MLTPLX  $Q1MN   0       ACTIVE   YES        1                YES


 LOGICAL UNIT                 REQUEST  NB BLOCKS  STATUS   ERROR  ORIGIN
 NAME                         NAME     TRANSFERED

 \ETINET.$Q1R1.#Q1RMU05       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU06       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU07       NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU08       NONE                OUTSTND  0      NONE
```

Example 6 - Status of Remote-location

The following STATUS command displays the status of all active requesters associated To remote-location %Q1M.

```
~status %q1m
*** CURRENT STATUS AT 2001-08-06  14:23:18

  PROCESS             JOBS                 MAX    ENABLE    DATA
TYPE    ID      SELECT  STATUS   CONNECTED RQSTS  COMPRESS  SORTED  ENCRYPT
STNDRD  $Q1MR   0       ACTIVE   YES       2      YES       YES     NO
MLTPLX  $Q1MM   0       ACTIVE   YES       1      YES       YES     YES
```

Example 7: - Status of ETI-NET File Server

If all parameters are omitted from the command, then the status of the entire ETI-NET File Server environment will be displayed — including the local location.

```
~status

*** CURRENT STATUS AT 2001-08-06  14:31:12

 LOCAL      LU      RETRY      XFER    WINDOW    MAX     CPU
 LOCATION   RETRY   TIME       RETRY   COUNT     BLOCK   LIST
 %Q1T       10      1          5       50        15000   00,01


            BACKUP   DAILY-TASK                                CATALOG  TX
            CATALOG  TIME  FILE                                TIMER    SIZE
            NO       NONE  NONE                                10       1000

            NORMAL-IO-COMPLETIONS-CODES
            ALL

            FS SECURITY        MASTER   AUTO     LOGGER    EVENT    INACTIVE
            LEVEL              OPERATOR STARTUP  SHUTDOWN  LOGGING  TIMER
            0                  NONE     NO       YES       NO       5

            DNS-NAME-MANAGER-PROCESS  <* not defined *>
            LOGGER-PROCESS            \ETINET.$Q1LG
            BCOM-IO-READ              \ETINET.$SYSTEM.SYS00.FUP
            BCOM-IO-READX             \ETINET.$DATA3.BKV510E.BFSCPYX
            BCOM-IO-WRITE             \ETINET.$SYSTEM.SYS00.FUP
            BCOM-IO-WRITEX            \ETINET.$DATA3.BKV510E.BFSLOAD
            BCOM-IO-JOB               \ETINET.$SYSTEM.SYS00.TACL
            BCOM-IO-SQL               \ETINET.$SYSTEM.SYSTEM.SQLCI
            BCOM-IO-BACKUP            \ETINET.$DATA3.BKV510E.BFSBKUP
            BCOM-IO-RESTORE           \ETINET.$DATA3.BKV510E.BFSREST
            BCOM-TDM-BACKUP           \ETINET.$SYSTEM.SYS00.BACKUP
            BCOM-TDM-RESTORE          \ETINET.$SYSTEM.SYS00.RESTORE
            BCOM-TSM-STD-CLIENT       \ETINET.$DATA3.BKV510E.TSMREQ
            BCOM-APPC-STD-CLIENT      \ETINET.$DATA3.BKV510E.BFSREQ
            BCOM-APPC-MPLX-CLIENT     \ETINET.$DATA3.BKV510E.BFSREQX
            BCOM-FTP-STD-CLIENT       \ETINET.$DATA3.BKV510E.FTPREQO
            BCOM-APPC-STD-SERVER      \ETINET.$DATA3.BKV510E.BFSSER
            BCOM-APPC-MPLX-SERVER     \ETINET.$DATA3.BKV510E.BFSSERX
            BCOM-FTP-STD-SERVER       \ETINET.$DATA3.BKV510E.FTPSERO
            BCOM-EP-STD-CLIENT        \ETINET.$DATA3.BKV510E.BCCLT
            BCOM-EP-STD-SERVER        \ETINET.$DATA3.BKV510E.BCSET
            BCOM-EP-STD-CHILDC        \ETINET.$DATA3.BKV510E.BCCCT
            BCOM-EP-STD-CHILDS        \ETINET.$DATA3.BKV510E.BCSET
            BCOM-EP-DIRSERVICE        \ETINET.$Q1DS
            BCOM-EP-CONFIG            \ETINET.$DATA.Q1TV510.BMCFG
            BCOM-EP-RESOURCE          \ETINET.$DATA.Q1TV510.BCR00
            BCOM-EP-INDEX             \ETINET.$DATA.Q1TV510.BCX00
            BCOMF-CATBK               \ETINET.$DATA.Q1TV510.CATBK
            BCOMF-CATBKFIL            \ETINET.$DATA.Q1TV510.CATBKFIL
            STATISTIC-FILE            \ETINET.$DATA.Q1TV510.STYMMDDN
            STARTUP-OBEY-FILE         <* not defined *>
            SHUTDOWN-OBEY-FILE        <* not defined *>
            QUEUE-FILE                $DATA.Q1TV510.QUEUE
            REQUEST-FILE              $DATA.Q1TV510.REQUEST


BCOM-REMOTE-IDENTIFICATION

   REMOTE     PLATFORM     SHORT     APPLID
   LOCATION   TYPE         FORM
   %Q1M       IBM:P        YES       Q1M
   TPN
   Q1M

APPC-REQUESTOR

   PROCESS NAME: $Q1MM        TYPE : MULTIPLX

   MAX   AUTO   ENABLE    PROCESS  IO-PROCESS   MAX    WINDOW DATA   ENCRYPT
   RQST  START  COMPRESS  CPU PRI  CPU      PRI  BLOCK  COUNT  SORTED
   1     NO     YES       0   140  SAME     -1  15970  50     YES    YES

   STARTUP DEFINES: <none defined>

ROUTING

   REMOTE    REMOTE             ROUTING
   LOCATION  APPLID             CLASS
   %Q1M      Q1M                2 3 4 5 6

   TPN
   Q1M

   LOGICAL UNIT NAME            DEST. LU NAME       DEST. MODE NAME
   \ETINET.$Q1R1.#Q1RMU15
   \ETINET.$Q1R1.#Q1RMU16
   \ETINET.$Q1R1.#Q1RMU17
   \ETINET.$Q1R1.#Q1RMU18
   \ETINET.$Q1R1.#Q1RMU19
   \ETINET.$Q1R1.#Q1RMU20

APPC-REQUESTOR

   PROCESS NAME: $Q1MR        TYPE : STANDARD

   MAX   AUTO   ENABLE    PROCESS  IO-PROCESS   MAX    WINDOW DATA   ENCRYPT
   RQST  START  COMPRESS  CPU PRI  CPU      PRI  BLOCK  COUNT  SORTED
   2     NO     YES       1   140  SAME     -1  4096   20     YES    NO

   STARTUP DEFINES: <none defined>

ROUTING

   REMOTE    REMOTE             ROUTING
   LOCATION  APPLID             CLASS
   %Q1M      Q1M                1 2 3 4 11 12 13 14 15

   TPN
   Q1M

   LOGICAL UNIT NAME            DEST. LU NAME       DEST. MODE NAME
   \ETINET.$Q1R1.#Q1RMU09
   \ETINET.$Q1R1.#Q1RMU10


EP-REQUESTOR

   PROCESS NAME: $Q1MA        TYPE : STANDARD

   MAX   AUTO   ENABLE    PROCESS  IO-PROCESS   MAX    WINDOW DATA   ENCRYPT
   RQST  START  COMPRESS  CPU PRI  CPU      PRI  BLOCK  COUNT  SORTED
   4     NO     NO        0   120  SAME     -1  32000  50     YES    NO

   STARTUP DEFINES: <none defined>

ROUTING

   REMOTE    ROUTING
   LOCATION  CLASS
   %Q1M      1 2 3 4 5

   END-POINT: EP.CLIENT    (2,2,4,30)
```

```
EP-SERVER

   PROCESS NAME: $Q1MB        TYPE : STANDARD

   MAX    DEST                        PRINT  AUTO   MAX     WINDOW  DATA
   RQST   FILE                        CNTL   START  BLOCK   COUNT   SORTED
   4      $S                          NO     NO     32000   50      YES

   OPT   PROCESS    IO-PROCESS
   FLAG  CPU  PRI  CPU      PRI
   0     0    120  SAME      -1

   STARTUP DEFINES: <none defined>
   END-POINT: SERVER.NSK   (4)


APPC-SERVER

   PROCESS NAME: $Q1MS        TYPE : STANDARD

   MAX    DEST                        PRINT  AUTO   MAX     WINDOW  DATA
   RQST   FILE                        CNTL   START  BLOCK   COUNT   SORTED
   4      $S.#Q1MS                    YES    NO     15970   50      YES

   OPT   PROCESS    IO-PROCESS
   FLAG  CPU  PRI  CPU      PRI
   0     0    140  SAME      -1

   STARTUP DEFINES: <none defined>

   LOGICAL UNIT NAME           DEST. LU NAME       DEST. MODE NAME
   \ETINET.$Q1R1.#Q1RMU01
   \ETINET.$Q1R1.#Q1RMU02
   \ETINET.$Q1R1.#Q1RMU03
   \ETINET.$Q1R1.#Q1RMU04

APPC-SERVER

   PROCESS NAME: $Q1MN        TYPE : MULTIPLX

   MAX    DEST                        PRINT  AUTO   MAX     WINDOW  DATA
   RQST   FILE                        CNTL   START  BLOCK   COUNT   SORTED
   1      $S                          NO     NO     15970   50      YES

   OPT   PROCESS    IO-PROCESS
   FLAG  CPU  PRI  CPU      PRI
   0     0    140  SAME      -1

   STARTUP DEFINES: <none defined>

   LOGICAL UNIT NAME           DEST. LU NAME       DEST. MODE NAME
   \ETINET.$Q1R1.#Q1RMU05
   \ETINET.$Q1R1.#Q1RMU06
   \ETINET.$Q1R1.#Q1RMU07
   \ETINET.$Q1R1.#Q1RMU08
```

Note — FTP option: *Examples of FTP requester and server processes are shown in the BCOM/FTP for NSK Guardian — Installation and User's Guide.*

Example 8: - Status of a request

The following will display the status of the request name #C1U002.

```
~ STATUS   #C1U002

*** CURRENT STATUS AT 2001-08-06  14:45:49

REQUEST NAME         = #C1U002
TYPE                 = SEND-FILE
QUEUE TIME           = 2001-06-20 16:10:22
START TIME           = 2001-06-20 16:10:51
END TIME             = 2001-06-20 16:13:02
QUEUE USER ID        = 130,255
RECORD COUNT         = 210
BLOCK COUNT          = 21
BFSCPYX  COMPL. CODE = 0
BFSCPYX  TERM.       = <None>
JOB-ID               = $X3W9
JOB-OBJ              = FUP
MESSAGE              = TRANSFER SUCCESSFUL
```

The following will display the status of a backup request.

```
~STATUS #C1U003

*** CURRENT STATUS AT 2001-08-06  14:46:41

REQUEST NAME         = #C1U003
TYPE                 = FILE-BACKUP
QUEUE TIME           = 2001-06-20 16:53:30
START TIME           = 2001-06-20 16:54:00
END TIME             = 2001-06-20 16:54:09
QUEUE USER ID        = 130,255
RECORD COUNT         = 307
BLOCK COUNT          = 216
BFSBKUP  COMPL. CODE = 0
BFSBKUP  TERM.       = <None>
JOB-ID               = $X3XJ
JOB-OBJ              = BFSBKUP
MESSAGE              = TRANSFER SUCCESSFUL
```

Note that Routing-class and Priority-class are no more displayed after the request has ended.

Other possible values in the MESSAGE = field are as follows:

```
abnormal termination
aborting
awaiting completion
deleted by operator
executing
file allocation error
hold state
open error
retry state
selected
transfer error
waiting
```

Explanations for Output Fields

The output fields for all STATUS command examples above are explained below.

For the BCOM File server STATUS command (i.e. STATUS with no parameter), the module names that are shown are the program-object file names that come with the BCOM install. Should the log process be inoperative, the error number received will be displayed on the second line, and logging will take place to $0.

ABORTING
indicates that this request was the object of an ABORT command.

AWAITING COMPLETION
indicates that BCOM is waiting for acknowledgement of its close request from its remote partner.

BCOM-APPC-MPLX-CLIENT
The BCOM client module that initiates multiplexed transfer requests from this local NSK Guardian node.

BCOM-APPC-MPLX-SERVER
The BCOM server module that services multiplexed transfer requests initiated from a remote location.

BCOM-APPC-STD-CLIENT
The BCOM client module that initiates transfers from this local NSK Guardian.

BCOM-APPC-STD-SERVER
The BCOM server module that services an inbound request for transfers that can be RF, SF, RJ, etc.

BCOM-EP-STD-CLIENT
The BCOM-EP client (Parent) module that initiates transfers from this local NSK Guardian.

BCOM-EP-STD-SERVER
The BCOM-EP server (Parent) module that services an inbound request for transfers that can be RF, SF, RJ, etc.

BCOM-EP-STD-CHILDC
The BCOM-EP client (Child) module that initiates transfers from this local NSK Guardian.

BCOM-EP-STD-CHILDS
The BCOM-EP server (Child)module that services an inbound request for transfers that can be RF, SF, RJ, etc.

BCOM-EP-DIRSERVICE
The BCOM-EP Directory services module.

BCOM-EP-CONFIG
The BCOM-EP Configuration file.

BCOM-EP-RESOURCE
The BCOM-EP Directory service file.

BCOM-EP-INDEX
The BCOM-EP Directory service (index) file.

BCOM-FTP-STD- CLIENT
The BCOM FTP client module that initiates transfers from this local NSK Guardian.

BCOM-FTP-STD- SERVER
The BCOM FTP server module that services an inbound request for transfers that can be RF, SF, RJ, etc.

BCOM-IO-BACKUP
BBACKUP is the module used to initiate the backup of NSK Guardian files to an MVS site. The data is transferred using the standard BCOM transfer process.

BCOM-IO-JOB
The command interpreter module (TACL) which implements Obey file processing on NSK Guardian platforms.

BCOM-IO-READ
This file-transfer module (FUP) is responsible for transferring records efficiently from NSK Guardian Enscribe structured files into BCOM for transmission.

BCOM-IO-READX
This file-transfer module (BCOPYX) is responsible for transferring efficiently records from NSK Guardian Enscribe unstructured files, processes, devices into BCOM for transmission.

BCOM-IO-RESTORE
BRESTORE is the module used to restore data information previously saved on the MVS site through a backup procedure.

BCOM-IO-SQL
This file-transfer module is responsible for transferring efficiently records to and from NSK Guardian SQL tables.

BCOM-IO-WRITE
This file-transfer module (FUP) is responsible for efficiently transferring records from BCOM into NSK Guardian Enscribe structured files.

BCOM-IO-WRITEX
This file-transfer module (BFSLOAD) is responsible for transferring efficiently records from BCOM into NSK Guardian Enscribe unstructured files, processes, devices.

BCOM-TDM-BACKUP
the NSK Guardian utility (BACKUP) that is used by BCOM to take a backup of NSK Guardian disc files for transmission to a remote site.

BCOM-TDM-RESTORE
the NSK Guardian utility (RESTORE) that is used by BCOM to reload onto NSK Guardian discs, the data files transmitted from a remote site.

BCOM-TSM-STD-CLIENT
The BCOM TSM client module that initiates transfers from this local NSK Guardian.

BCOMF-CATBK
The BCOM Backup Local Catalogue.

BCOMF-CATBKFIL
The BCOM Backup Local Catalogue.

CONNECTED
indicates whether this transfer process is currently connected with the target remote under which it is running. YES indicates that it is. A STATUS on the process name will result in the display of all the LU sessions under the control of this transfer process.

ENABLE-COMPRESS:
This process can handle the request for compression, provided that the user has asked for it in his request [Reference: request <YES | NO> of the request definition]

ENCRYPTION:
This parameter, which is used to enable the BCOM's ENCRYPTION facility, can be set in three different types of BCOM Configuration file's entity: the BCOM's main process definition, the requester process definition and the routing group definition.

ERROR
will display the error code if BCOM receives an error in response to an I/O to the device.

FILE ALLOCATION ERROR
is returned when the remote partner is unable to allocate the target dataset. For example, if the partner is an MVS, conflicting DCB information would cause this error.

HOLD STATE

indicates that this request was the object of a QUEUEHOLD command.

If the EMS option was chosen at installation time, all messages will be sent to the $0 area.

In a requester process definition, any value of this parameter (YES or NO) overrides the value of the same parameter for the main process and defines the default value for the same parameter in all routing group definitions inside this same process definition.

In a routing group definition, any value of this parameter (YES or NO) overrides the value of the same parameter for the requester process where this routing group is referred to. The value of a routing group's ENCRYPTION parameter will determine whether or not the ENCRYPTION facility will be used over lines defined in this routing group.

In the main process definition, this parameter defines (YES or NO) the default value for the same parameter in all requester process definitions. Default is NO.

io-process COMPL. CODE = nn

displays the name of the IO process program and its completion code after the execution of the request.

io-process TERM. = xxxxxxxxxxxxxxxxxxxxxxxxx

displays the possible text received in the completion information of the IO-process

JOB STATUS

indicates the STATUS of the transfer process being displayed. Possible values are DRAINING or PROCESSING. If a transfer request has been ABORTed it will not show up as a line in the STATUS display. Only currently processing and draining transfer-processes are displayed in the STATUS command.

JOBS SELECT

indicates the number of requests accepted by this transfer process that have been assigned an LU-LU session and are currently awaiting acknowledgement from the remote partner that the target dataset has been allocated.

LAST CHECKPOINT

In a file transfer request, displays the record count of the last checkpoint.

In a backup request, displays the number of files and the last Guardian filename included in the last checkpoint.

LAST FILE NAME

for a backup or restore request, displays the last Guardian file name seen in the backup data-stream.

LOGGER PROCESS

This is the logger process supplied with the BCOM product. The logger logs to two disk files; once a first disk is full, the other is cleared and written to. If a critical error is detected within the process, $0 is used as an alternate logging location.

LOGICAL UNIT NAME:

SNA communication drive used to transfer your data to the remote location. This LU name is defined to the SNA access method (SNAX/XF, SNAX/CDF or SNAXLINK) using the NSK Guardian utility CMI or SCF, and to BCOM FS 6.2 in the BCOM Configuration file.

MAX-RQSTS

is the value specified in the Configuration file which controls the maximum number of transfer requests that this transfer process will take on at any point in time. MAX-RQSTS can only be as high as the total count of LUs configured under this transfer process.

NB BLOCKS TRANSFERRED / BLOCK COUNT

is blocks (as opposed to records) transferred, and displays the current transfer count for jobs executing.

OPEN-ERROR

If the remote partner's dataset can be allocated but not opened this error will be displayed.

ORIGIN

is a field that contains the name of the sub-system from which the ERROR was issued. Possible values are GUARDIAN or SNA. This will allow interpretation of the ERROR field value.

PROCESS ID

indicates the name of the process, as entered in the STATUS command.

QUEUE FILE

ENSCRIBE key sequenced file that contains the execution instances of transfer requests defined in the Request file.

RECORD COUNT

is the number of transferred records

REMOTE LOCATION

indicates the name of the remote location as BCOM recognizes it.

REQUEST FILE

ENSCRIBE key sequenced file that defines BCOM transfer requests, including all their transfer attributes and completion statistics from the last execution.

REQUEST NAME:
Name of a BCOM request currently being serviced on the LU. If there is no current session, this field will have a value of "NONE".

RETRY STATE
indicates that BCOM has detected an error in a transfer that can be retried, and is beginning to retry this transfer.

ROUTING CLASS
indicates the class (type of connection) to which local requests are queued in order to be sent to the remote location.

ROUTING TYPE
indicates whether this transfer process is STNDRD or MLTPLX. MLTPLX indicates that any transfer requests assigned to the routing-class containing this transfer-process will utilize all available LUs in a round-robin fashion. Refer to Section 5 for a description of the multiplexing option.

SELECTED
indicates that BCOM has selected this request for initiation and is currently waiting on the remote partner's reply to the allocation message.

SESSION ID:
Not used.

SHUTDOWN-OBEY-FILE
Customizing Obey file at BCOM shutdown

STARTUP-OBEY-FILE
Customizing Obey file at BCOM startup

STATISTIC-FILE
The BCOM Request transfer Statistics file.

STATUS
normally displays "OUTSTND" indicating that there is at least 1 outstanding I/O operation for LU device identified on the same output line.

TIME INTERVAL
defined through the configuration parameter LRETIME, it determines the period between automatic attempts at re-connecting / re-starting transfer processes.

TRANSFER-ERROR
indicates that the transfer request has started and has encountered a transmission error. If checkpoint has been specified, this request can be restarted.

WAITING
is the status when a request has been queued but has not yet been SELECTED for execution.

SWITCH STATS

Function:

To close the current Statistics file and open a new one.

Syntax:

```
SWITCH STATS
```

Considerations:

1. This command is available ONLY for the Master Operator.
2. The new Statistic file is created at the next request transfer completion.
3. If BFSMON was not started with the =BCOM-STATS-FILE DEFINE, the following warning message I s issued:
   ```
   "STATISTIC FILE IS NOT DEFINED."
   ```

Example:

The SWITCH STATS close the current Statistic file.

```
~switch stats
STATISTIC FILE IS SWITCHED.
```

TSM

Function:

This command introduces a subcommand for BackHome/TSM option.

Syntax:

```
TSM subcommand parameter
```

Considerations:

Refer to the *BackHome/TSM - Installation and User Guide* for details.

UNCATALOGBACKUP (Obsolete)

Function:

To remove from the local backup catalog, all entries related to a backup execution

Syntax:

```
UNCATALOGBACKUP | UNCATBK remote-object-name
```

where

`remote-object-name`	MVS file name or TSM object name, depending on the server type where the backup was stored. A wildcard (**`*`**) can end the remote-object-name.

Considerations:

This command is no longer supported.

WARMSTART

Function:

This command causes a transfer process to attempt to re-establish sessions with its remote partner for all LUs (or IP sessions) configured under this transfer process.

Syntax:

```
WARMSTART {<process-id> | ALL}
```

where:

process-id	is the name of an active BCOM transfer process.
ALL	means all active BCOM transfer processes.

Considerations:

- This command is available ONLY for the Master Operator.
- This command may be used if, after doing a STATUS command on a transfer process, the user sees that the transfer process is not CONNECTED.
- transfer process must be running in order to execute a WARMSTART command against this process.
- A transfer process will show CONNECTED if at least one LU has a STATUS of ACTIVE.
- WARMSTART may be used to activate the LUs (or IP sessions) of a transfer process that are not ACTIVE.
- If the LINE, PU or LU are not active at the time that transfer processes are STARTed, WARMSTART can be used to re-establish the LU sessions after LINE, PU or LU have been started.
- the WARMSTART command resets the RETRY counter to zero.

Example 1:

```
~ WAMSTART ALL
PROCESS $Q1MR WARMSTARTED
PROCESS $Q1MS WAMRSTARTED
```

Example 2:

The following command forces the $Q1MR process to reconnect with the MVS BCOM program.

```
~status $q1mr
.
*** CURRENT STATUS AT 2001-08-06  15:09:30

  PROCESS           JOBS                  MAX    ENABLE    DATA
TYPE    ID      SELECT  STATUS   CONNECTED RQSTS  COMPRESS  SORTED  ENCRYPT
STNDRD  $Q1MR   0       ACTIVE   NO        2      YES       YES     NO

ROUTING

 REMOTE    REMOTE              ROUTING
 LOCATION  APPLID              CLASS

 %Q1M      Q1M                 1 2 3 4 11 12 13 14 15

 TPN
 Q1M

 ENCRYPTION: NO

 LOGICAL UNIT                  REQUEST  NB BLOCKS  STATUS   ERROR  ORIGIN
 NAME                          NAME     TRANSFERED

 \ETINET.$Q1R1.#Q1RMU09        NONE                PROCSSNG H0857  SNAX
 \ETINET.$Q1R1.#Q1RMU10        NONE                PROCSSNG H0857  SNAX


~warmstart $q1mr
PROCESS $Q1MR  WARMSTARTED.


~status $q1mr
*** CURRENT STATUS AT 2001-08-06  15:09:41

  PROCESS           JOBS                  MAX    ENABLE    DATA
TYPE    ID      SELECT  STATUS   CONNECTED RQSTS  COMPRESS  SORTED  ENCRYPT
STNDRD  $Q1MR   0       ACTIVE   YES       2      YES       YES     NO

ROUTING

 REMOTE    REMOTE              ROUTING
 LOCATION  APPLID              CLASS

 %Q1M      Q1M                 1 2 3 4 11 12 13 14 15

 TPN
 Q1M

 ENCRYPTION: NO

 LOGICAL UNIT                  REQUEST  NB BLOCKS  STATUS   ERROR  ORIGIN
 NAME                          NAME     TRANSFERED

 \ETINET.$Q1R1.#Q1RMU09        NONE                OUTSTND  0      NONE
 \ETINET.$Q1R1.#Q1RMU10        NONE                OUTSTND  0      NONE
```

THIS PAGE WAS LEFT BLANK INTENTIONALLY.