BCOM for MVS
Version 5

User's Guide

BackHome, BCOM and HCOM are trademarks of ETI-NET

ISSUE NUMBER: UGD-BM-V500-033

CREATION DATE: SEPTEMBER 1999

REVISION DATE: 2005-07-14

Printed in Canada

ETI-NET, 180 Rene-Levesque blvd. East, suite 320, Montreal, Quebec, Canada, H2X 1N6
Tel.: (514)395-1200
Fax: (514)395-9080

Table of contents

Preface

Readers of the *BCOM for MVS User's Guide* are expected to be already familiar with the basic concepts and facilities of BCOM in order to make good use of this manual. For completeness sake, however, this manual introduces the general concepts and features of BCOM before inviting the reader to define and control his or her transfer requests.

A command syntax and reference manual is provided in *Appendix A*. The knowledgeable user should be able to use this reference section as a quick reminder of the possibilities for each of the BCOM commands.

Related BCOM publications

BCOM for MVS – Installation and Operations Guide

BCOM for NSK Guardian – Installation and Operations Guide

BCOM for NSK Guardian – User's Guide

BCOM EP – Installation and Migration Guide

AnyPrint! – User Guide

ETI-NET Messages Manual

Presenting BCOM Facalities

This section introduces the reader to the basic Concepts and Facilities of the BCOM product, the terminology in use, the features and the various user interfaces.

This section covers the following topics:

- What Is BCOM? (General Description)
- BCOM Terminology
- BCOM Standard Features
- BCOM Platform Specific Features
- BCOM Product Support

What is BCOM?

This manual discusses all the aspects involved in using BCOM on an MVS node.

BCOM provides file transfer, report distribution, and job submission between different or similar hardware and operating systems using a peer-to-peer protocol.

The diagram on the next page illustrates how BCOM is used to integrate your network.

What BCOM does?

ETINET BCOM software solves these problems by allowing efficient, bi-directional file transfers to take place between several popular platforms - including: MVS on IBM/Mainframe, NSK Guardian on Tandem, and Windows NT.

The BCOM product allows for the efficient management of file transfer, report distribution and job submission across environments. This product will reduce network and manpower costs and increase batch window throughput.

It sends and receives print and job files between cooperating applications, allowing for standard application jobs and for processes. In addition, several methods of data transfer can be selected through a friendly user interface. All transfers take place across SNA lines or devices (including the channel-attached SNAXLINK for NSK Guardian to MVS), or across TCP/IP connection.

Transfer definition and activation characteristics are totally under the user's control. Once a transfer has been defined, it can be re-activated any number of times. Specific instructions allow the control of ASCII-EBCDIC field conversion, and jobs can be submitted from either side.

BCOM uses a variety of blocking techniques to ensure efficient file transfer rates. For large file transfers, or for transfers over low speed lines, checkpoint and restart facilities are provided. In addition, a large file can be multiplexed, that is, sent over multiple lines at the same time.

The BCOM product supports all major file types on all platforms. Also supported is interactive or batch mode file transfer definition or activation on any of the eligible nodes. Transfer activation can

be integrated into general job scheduling software within the architecture of participating systems.

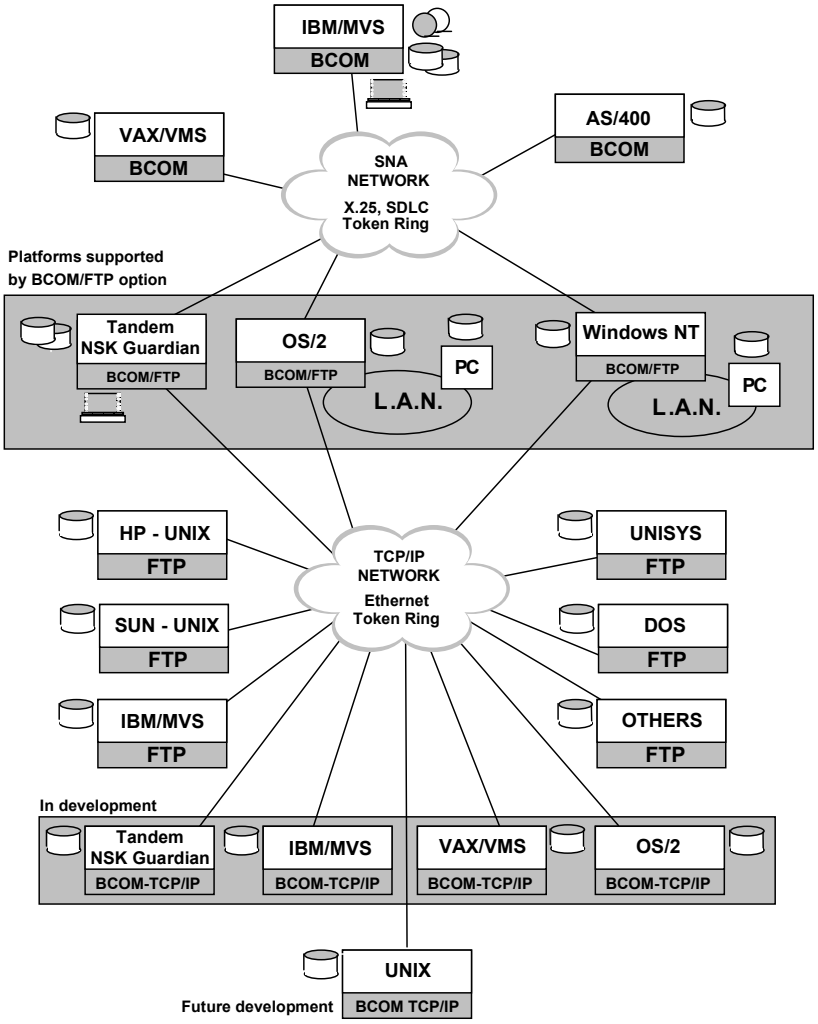
With BCOM, automatic follow-on job scheduling can be set up across environments; this means that jobs can be chained together, with each job starting automatically based on the termination status of the previous one. Take note that BCOM now supports automatic follow-on scheduling that can include local jobs using local job data files.

A security feature can also be used to cause BCOM on all platforms to access system data, files and resources under the security profile of the user that initiates or queues a BCOM transfer.

An optional security feature can also be used to restrict the use of critical operations to privileged users.

For a complete list of BCOM features, refer to BCOM Features, later in this section.

Figure 2-1: The BCOM Network



BCOM Terminology

This section introduces the terminology used in this guide and throughout the BCOM manuals.

A grasp of the basic meaning of these terms is essential to a proper understanding of BCOM.

This section is simply an introduction to these terms. More detailed and advanced information about the BCOM functions is provided later in this guide and in the *BCOM Installation and Operations Guide* for the appropriate platform.

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, the request can be queued for execution.

It should be noted that print jobs originating on a NSK Guardian spooler or MVS need not be defined as requests to BCOM: definition and queuing of these job types are automatic.

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

BINT is the acronym used to describe the BCOM User Interface. This interface can be used in both batch and interactive mode on all supported platforms. The BINT interface - also called B62BINT - is the vehicle through which all requests are defined and queued for the BCOM nucleus. In addition, all operational control commands are entered through B62BINT.

Local vs 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 B62BINT program executes and wherein a transfer request gets to be recorded into the Request file. When that node initiates a transfer to another node, the latter is called a REMOTE LOCATION.

The Requester Node

The requester node is the local or remote node which 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, which is used to initiate transfers from the local location.

The Server Node

The server node is the node which 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 or 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 be initialized 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 Routing Class

As explained earlier, each local BCOM node defines both the LOCAL LOCATION attributes and some REMOTE LOCATION attributes.

Each of these remote locations can be accessed through multiple routing classes. A routing class defines outbound path control information for sending transfer requests to the remote location. The user can define several Logical Units or parallel sessions within the same routing class.

The routing class is also characterized by a routing type. BCOM allows standard and multiplexed transfer types.

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 Submission across Environments

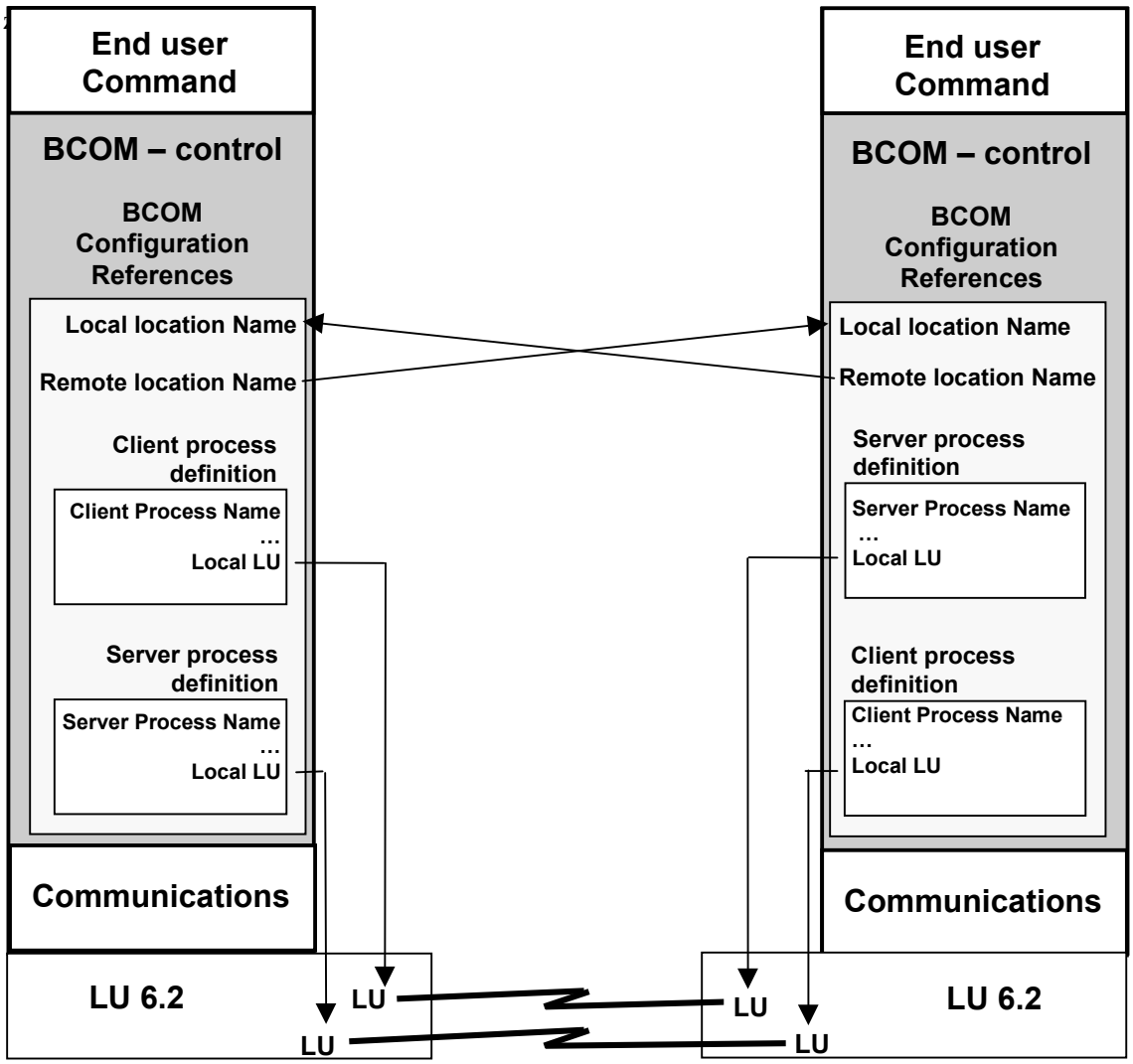
BCOM allows a local node to submit jobs to any of its target nodes. Support is available for NSK Guardian's TACL, MVS's JCL, and Windows NT's .exe and .bat.

BCOM for NSK Guardian supports WAITED job submissions. OBEY type jobs can be submitted from a remote site with WAITED option: this will cause the request to terminate only after the completion of the job.

Layered Architecture

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

Figure 2-2: BCOM Architecture



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 requests can be held, removed or released.

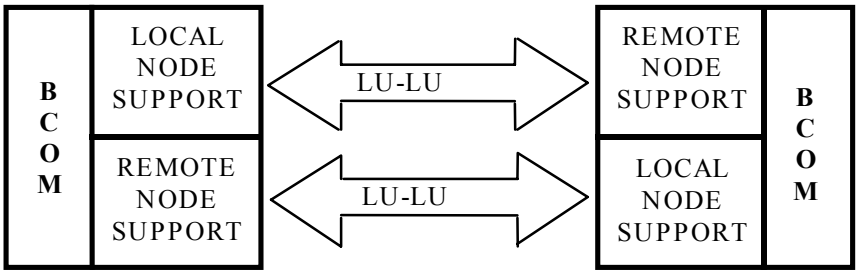
Report Distribution Across Environments

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

SNA LU 6.2 Symmetric Architecture

BCOM is fully designed using the SNA LU 6.2 peer-to-peer protocol. It offers bi-directional and concurrent inter-platform data transfers over a BCOM controlled LU to LU session. It exploits the buffered send techniques of BCOM to provide efficient data transfers.

Figure 2-3.



Each BCOM node acts as both a local and remote partner. BCOM is a local node for the purposes of anyone on that platform; but it is also a remote node to the people on the other platforms.

User Defined Checkpoint/Restart

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

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.

Auto Re-START 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 and then automatically re-start the request from the beginning.

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.

User Exit Routines

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

MVS-Specific BCOM Features

In addition to the above features, BCOM for MVS Version 4 or later offers the following:

Automatic Restart of Abended MVS Transfer Subtasks

A separate subtask is attached for each initiated transfer. BCOM automatically restarts an executing subtask when it fails. Sub-task abends are trapped and an appropriate message generated.

Multi-Task Load Balancing

MVS subtasks started for BCOM can take advantage of the multiple CPUs in a loosely or tightly coupled processor environment.

Security

BCOM security facility provides users with the ability to secure access to the facilities of BCOM and to the files that could be accessed through BCOM, either on the local or remote platforms.

This security system either makes use of the platform residing security facility, to control access to files, or it can use a special BCOM security file inside which user-defined access profiles are maintained.

For controlling accesses of users from remote platforms, the BCOM security system makes use of a BCOM security table that correlates the User Identifications (User-Ids) of these foreign users with local User IDs.

The BCOM security system can also be adapted to the BCOM/FTP option. In such case, local User Ids will be correlated to remote FTP logins and passwords where required.

An IBM-ISPF Table Manager provides the security files on MVS and management/distribution of such files for other BCOM platforms.

ISPF Menu

The BCOM User Interface - called BINT - is used to communicate with an active BCOM Monitor (or Main Task). It is the vehicle through which all requests are defined and queued for transfer. In addition, all operational commands are entered through that interface.

Application Interface to CICS

The BCOM Application Programmatic Interface (API) for CICS is a transactional interface for on-line CICS transactions to define and queue requests programmatically to BCOM.

The API's connection relies on APPC/LU6.2 protocols and parallel sessions to dramatically simplify your configuration requirements: it lets you concentrate on the BINT functions while the CICS and BCOM sub-systems automatically handle the allocation and management of multiple, concurrent conversations.

MVS-API

The BCOM Application Programmatic Interface (API) for MVS is a callable interface for COBOL or Assembler programs to define and queue requests programmatically to BCOM.

PASSTHROUGH option

BCOM on MVS offers as an option a Passthrough facility, which allows the transferof data between two BCOM platforms that are not directly connected, but are both connected to a same MVS platform. With this utility, sending data is done between origin and destination platforms, the same way than if there was no intermediate platform, though it provides to the MVS administrator all privileges for controlling the use of the platform's resources. Please refer to the PASSTHROUGH Option of BCOM for MVS Installation and User's Guide for all details on this option.

Supported File Types

All VSAM (KSDS, ESDS, RRDS), QSAM, GDG, PDS and Tape files are supported on MVS.

When the remote location is an NSK Guardian, file data is accessed in the following fashion. 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).

Time Scheduling Facility

The BCOM Time Scheduling Facility (TSF) is a feature that allows you to set a specific date and time for your request, or automatically schedule your requests at fixed, repetitive day or hour intervals.

BCOM will keep (even across restarts) monitoring the queue file for TSF entries until their scheduled time has arrived.

Remote File Backup to MVS

BBACKUP is an option that enables you to copy or archive files to MVS from any supported remote BCOM platform, and to restore those files any time. With this option, you can back up files either to an MVS sequential file or directly to tape. This allows a remote network to make use of existing MVS tape management systems and procedures.

Resource Naming Conventions

Equally important to the BCOM command syntax is the naming convention used in designating requests, process IDs and locations.

The names used to specify these three types of resources must begin with a special character:

- Request-names start with a # sign;
- Location-names start with a % sign;
- Process-names start with a $ sign.

ETI-NET Customer and Software Support

ETI-NET provides various software support facilities, including maintenance, education and customization services. If you require special software services, ETI-NET will be happy to evaluate your needs.

A maintenance agreement for ETI-NET BCOM is available and is recommended for product upgrading.

ETI-NET also maintains a telephone Hot Line to respond to your questions. To contact us, send us an Email at support1@etinet.com or call (514) 395-1200 ext 336.

BINT - BCOM User Interface

With BCOM, data transfers are executed via a user interface: the BINT. This interface has been designed to receive and execute directly a user's commands, should these be defining or queuing requests. Also, BCOM allows the user to avoid repetitive long commands, each time he defines and queues a request.

There are essentially two ways to use the BINT, which are:

- The Interactive mode;
- The Batch mode.

The BINT can be used by either a user or a program.

This gives us four possible combinations, which are described in the following table:

INITIATIOR	MODE	
	Interactive	Batch
User	Command-line Mode	Batch Mode
Program	API Mode	Program Mode

The following figures illustrate these four approaches.

Figure 3-1: The Command-line Mode (Interactive)

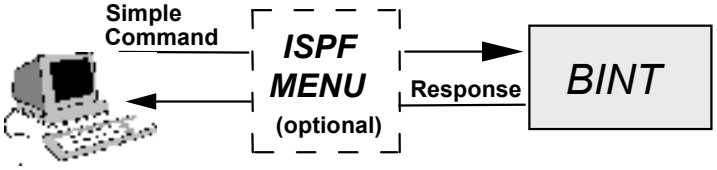
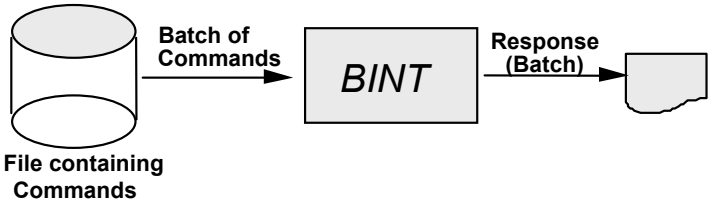


Figure 3-2: The Batch Mode (No Interaction)



In these two examples, a user initiates the process.

Figure 3-3: The Program Mode (No Interaction)

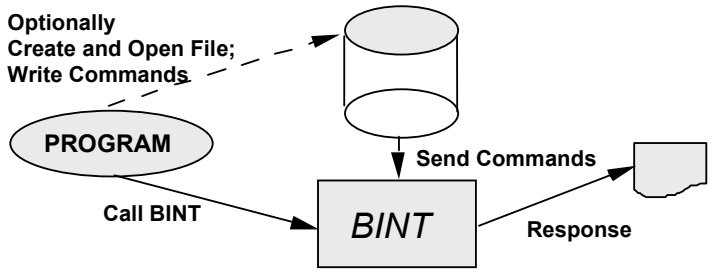
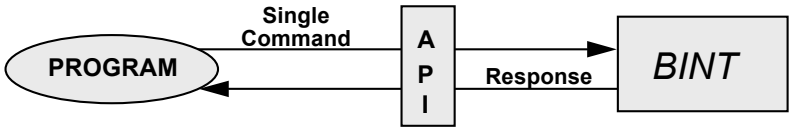


Figure 3-4: The API Mode (Interactive)



In these two examples, the process is initiated by a program or an instruction within a program. In the second example, the program receives a response after each sending of a single command and defines the next command depending on the nature of the preceding response.

Command-Line Mode

This method (see Figure 3-1), which is similar to an interactive session with the TSO based monitor, allows you to establish an interactive line mode session with the BINT module.

To invoke BINT in the interactive mode, log on to TSO an then enter the CLIST command, which will invoke the BINT. The CLIST, which is documented in the *BCOM for MVS - Installation and Operations Guide*, allocates the required DDnames and invokes the BINT program from the BCOM load library.

```
FREE     FILE  (COMMOUT,COMMIN)
ALLOCATE FILE(COMMOUT)     DA(*)
ALLOCATE FILE(COMMIN)      DA(*)
CALL 'your.MVS.loadid(B62BINT) <B62BINT-APPLID>,
   <b62main-APPLID>,& <def-Lloc>,
   &<def-Rtc>,<def-Pri>'
```

Note: *Insure that there are NO sequence numbers within the JCL files. (Use UNNUM command)*

The default parameters specified on the PARM=operand are used in conjunction with the SET and RESET commands of the BINT subtask. They will pre-set the corresponding keywords of SET commands and will also be used at RESET time. To override them, enter another value with the SET command.

Batch Mode

This execution mode (see Figure 3-2) allows you to batch together a series of BINT commands in a file and have them processed by BCOM.

To invoke BINT in batch mode, use the following JCL - which is also documented in the *BCOM for MVS - Installation and Operations Guide*:

```
//STEPxx  EXEC  PGM=B62BINT,
//parm='<B62BINT-APPLID>,<b62main-APPLID>,         X
//       <def-Lloc>,<def-Rloc>,                    X
//       <def-Rtc>,<def-Pri>'
//STEPLIB   DD  DISP=SHR,DSN=your.MVS.loadlib
//*
//SYSUDUMP  DD  SYSOUT=*
//COMMOUT   DD  SYSOUT=*
//COMMIN    DD  *
Your B62BINT commands
/*
```

The following parameters are provided:

B62BINT-APPLID mandatory; a VTAM APPLID to be used by your B62BINT subtask;

B62MAIN-APPLID mandatory; the VTAM APPLID of the B62MAIN task;

DEF-LLOC Optional; a default name for the local-location for request definitions; must begin with a % sign;

DEF-RLOC Optional; a default name for a remote-location for request definitions; must begin with a % sign;

DEF-RTC Optional; a default routing class for request definitions; value 00-99; must correspond to one of the routing class values defined under the remote-location in the configuration file;

DEF-PRI Optional; default priority for request definitions

Note1: *The optional parameters are positional.*

Example 1: ',B62MAIN-APPLID'

Example 2: ',B62MAIN-APPLID,DEF-LLOC,,DEF-RTC'

Note 2: *If you use sequence numbers within the JCL file, make sure to use the JCL option Facility described in the following pages. Without this option, sequence numbers would be interpreted with the B62BINT commands and create a syntax error.*

Options parameter

A parameter in the BINT EXEC can be used to define options when running in batch mode.

JCL Facility

The JCL facility changes the length of the input buffer for the commands, from 80 to 72 bytes. The result is that the "EDIT NUMBERS" are not scanned when the command is read.

Return code Facility

When the Return code Facility is OFF, the maximum return code of the BINT JCL is 00, 12 or 20, as in the standard BINT JCL. However, if the Return code Facility is ON, the maximum return code 04 and 08 are also possible, in addition to 00, 12 and 20. The complete list of BINT JCL return codes is as follows:

00 execution successful
04 warning; an event occurred during the execution of a BINT command
08 error; an error occurred during the execution of a BINT command
12 error; a communication error occurred between the BINT and BCOM
20 error; an error occurred during a QWAIT

Note: ***Reset return code to 00***
It is possible to reset the maximum return code to 00. For example, the user wants to ignore a return code for an error condition which is considered successful. The command syntax for this BINT command is:

```
RESET MAXRC
```

Options parameter syntax

The options parameter can be added to the PARM member of the B62BINT EXEC statement.

Two values are allowed for this parameter:

A means JCL Facility is ON and Return-code facility is ON

B means JCL Facility is OFF and Return-code facility is ON

Options parameter use examples

Here are three examples that show how to use or ignore the options parameter in a B62BINT EXEC statement.

Option A

```
//STEPxx EXEC PGM=B62BINT,
//       PARM='<b62bint-APPLID>,<b62main-APPLID>,        X
//       <def-lloc>,<def-rloc>,<def-rtc>,<def-pri>,A'
```

Option B

```
//STEPxx EXEC PGM=B62BINT,
//       PARM='<b62bint-APPLID>,<b62main-APPLID>,        X
//       <def-lloc>,<def-rloc>,<def-rtc>,<def-pri>,B'
```

No option

```
//STEPxx EXEC PGM=B62BINT,
//       PARM='<b62bint-APPLID>,<b62main-APPLID>,        X
//       <def-lloc>,<def-rloc>,<def-rtc>,<def-pri>'
```

Program Mode

This section describes how an MVS program could execute BINT commands (see Figure 3-3). The commands are entered on a file and the BINT is invoked programmatically through a CALL statement.

Step 1:

Write all BINT commands to the commin file, which must have the following characteristics:

- DDNAME=COMMIN
- LRECL=80
- RECFM=FB
- QSAM FILE

All output will go to the commout file, which must have the following characteristics.

- DDNAME=COMMOUT
- LRECL=80
- RECFM=FB
- QSAM FILE

Step2:

Close the above files.

Step 3:

Call the B62BINT subroutine as follows:

```
CALL "B62BINT" USING PARAM-INPUT
```

PARAM-INPUT must be defined as follows:

```
01 PARAM-INPUT.
05 PARAM-LEN    PIC 9(4) COMP VALUE 41.
05 APPLID-BINT  PIC X(8) VALUE 'xxxxxxxx'.
05 FILLER       PIC X    VALUE ','.
05 APPLID-BCOM  PIC X(8) VALUE 'yyyyyyyy'.
05 FILLER       PIC X    VALUE ','.
05 DEF-LLOC     PIC X(8) VALUE '%zzzzzzz'.
05 FILLER       PIC X    VALUE ','.
05 DEF-RLOC     PIC X(8) VALUE '%aaaaaaa'.
05 FILLER       PIC X    VALUE ','.
05 DEF-RTL      PIC X(2) VALUE 'bb'.
05 FILLER       PIC X    VALUE ','.
05 DEF-PRI      PIC X(2) VALUE 'cc'.
```

where:

'*xxxxxxxx*'	is the VTAM APPLID of your MVS BINT;
'*yyyyyyyy*'	is the VTAM APPLID of the BCOM program;
'*%zzzzzzz*'	is the default Local location name for request definitions;
'*%aaaaaaa*'	is the default Remote location name for request definitions;
'*bb*'	is the default Routing-class for request definitions;
'*cc*'	is the default priority class for request definitions.

If any problems are encountered in establishing communication with BCOM, B62BINT will return a condition code of 12.

API Mode

BCOM supports programmatic interfaces for MVS COBOL/Assembler programs and CICS on-line transaction programs.

BCOM CICS-API

The BCOM Application Programmatic Interface (API) to CICS is a transactional interface that allows users to define and queue requests programmatically from within on-line CICS transactions.

The CICS API is invoked by the issuing of an EXEC CICS LINK statement from within one of your transaction programs. The module of ETI-NET being called by this LINK statement will allocate a BINT conversation on the control session that has been established between CICS and BCOM at system initialization time.

The API's connection relies on APPC/LU6.2 protocols and parallel sessions to dramatically simplify any configuration requirements: it lets you concentrate on the BINT functions while the CICS and BCOM sub-systems automatically handle the allocation and management of multiple, concurrent conversations.

You can control the connection between CICS and BCOM from either side of the link - through BCOM operator commands or CICS operator commands, although the scope of control is not exactly the same.

The CICS-API programmatic interface is fully explained later in this manual.

BCOM MVS-API

The BCOM Application Programmatic Interface (API) for MVS is a callable interface for COBOL or Assembler programs to define and queue requests programmatically to BCOM.

The MVS API consists of a load module that must be link-edited to your program; using a defined interface convention, your program will cause this routine to allocate a BINT session with the Main task of BCOM and to process your BINT commands.

The API uses APPC/LU6.2 protocols to communicate with the Main Task of BCOM; it is designed to manage all aspects of session management with the BCOM sub-system such that your program can concentrate on the BINT functions at hand. A separate session is started for each MVS program that uses the API.

Meanwhile, operational commands are available to the BCOM operator in order to supervise or control the global connections of MVS programs to BCOM.

The MVS-API Programmatic Interface is fully explained later in this manual.

The BCOM ISPF Menu Interface

The BCOM User Interface - called BINT - is used to communicate with an active BCOM Monitor (or Main Task). It is the vehicle through which all requests are defined and queued for transfer. In addition, all operational commands are entered through that interface.

There are two methods by which you can interface to the BCOM program from your 3270 terminal:

- you can run the command-line interface from a TSO CLIST: this interactive method is based on a simple line-at-a-time display mechanism;
- you can run the Menu Interface designed for TSO-ISPF: this interactive method is based on panel displays and is a more congenial application for SAA users of 3270 terminals.

This chapter describes how to set up and use the Menu Interface of BCOM for TSO-ISPF.

Installation Considerations

The Menu interface for TSO-ISPF is shipped as a set of files on the BCOM Distribution Tape.

You can install the Menu Interface as a temporary, stand-alone CLIST, or you can integrate its panels to your TSO ISPF LOGON procedure for a more permanent installation.

For more information on how to install the Menu Interface, please refer to the ISPF-menu installation steps documented in the *BCOM for MVS - Installation and Operations Guide*.

Security

When you call the User Interface from within your TSO session, your LOGON USER ID is automatically captured and used for a number of internal functions - e.g.: establishing request ownership, accessing local resources, allowing a remote location, correlating your ID to another name which is local to that remote platform, etc.

For more information, please refer to *Appendix B* of the *BCOM for MVS - Installation and Operations Guide*.

Invoking the Menu Interface

The ISPF Menu Interface can be invoked two different ways, depending on prior selections.

Selecting from a PDF Menu

If the Primary option menu of your TSO's ISPF/PDF panels was modified to include a call to the BCOM Menu Interface, then simply go to that panel and type in the corresponding option.

Executing the Stand-Alone CLIST

If your CLIST library has been concatenated to your TSO-ISPF CLIST library at LOGON time, then simply type in the name of your CLIST from the ISPF option "6 Command":

```
MENUBINT
```

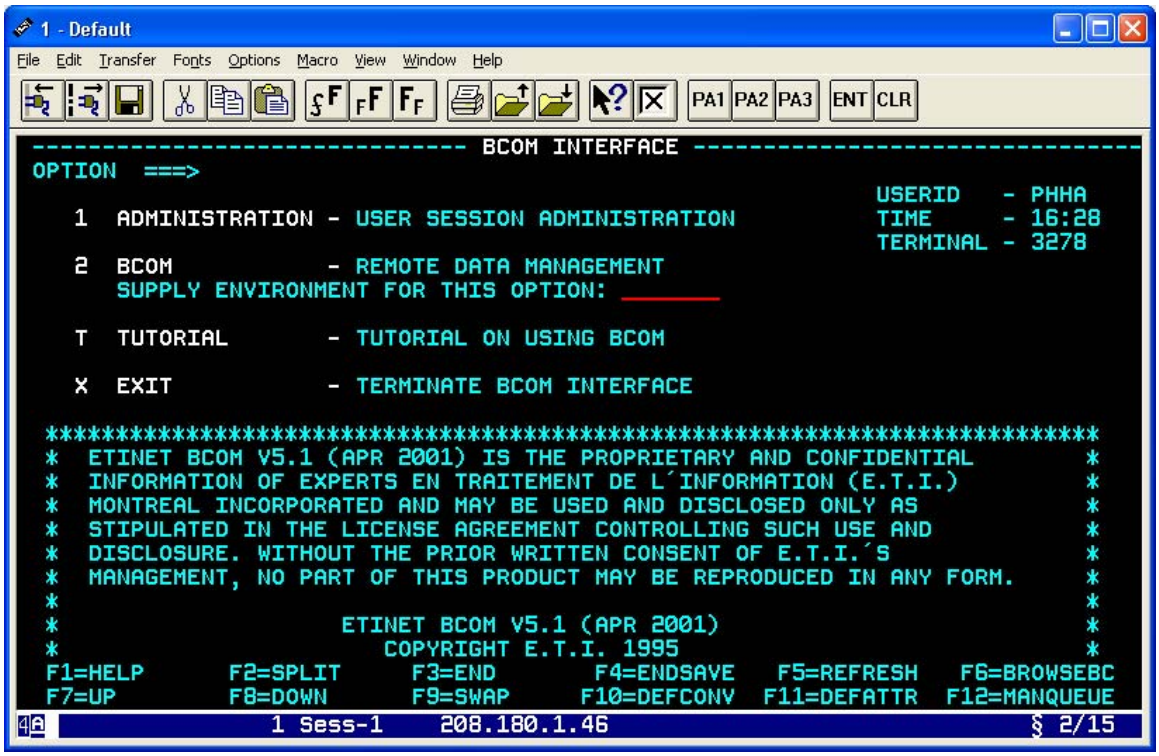
If you have created your CLIST library as a sequential CLIST dataset, or if your CLIST library is not dynamically allocated to SYSPROC, or if you are executing TSO in the command-line mode (outside of ISPF), then type the following fully qualified command name to invoke the Menu Interface:

```
EXEC   '<your.clist.dataset>(MENUBINT)'
```

The General Menu

When you start the Menu Interface, it greets you with an initial panel called the GENERAL MENU: this panel displays a copyright notice and lets you select some general function, as illustrated below:

Figure 4-1: The General Menu



General Menu Selections

The following options can be selected from the general menu:

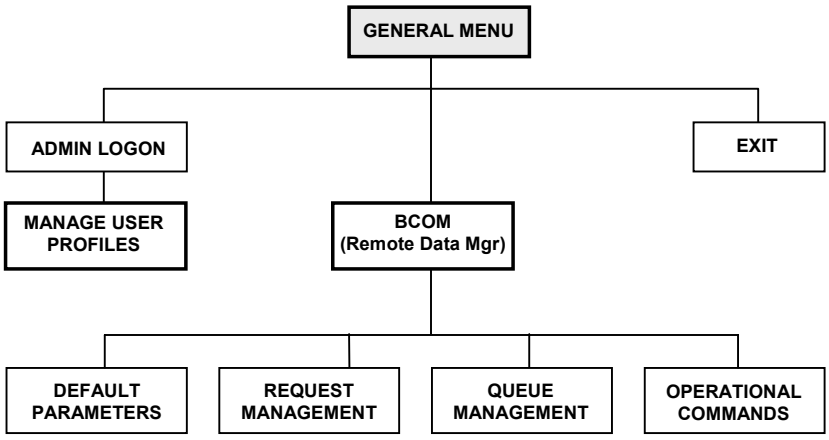
1	**ADMINISTRATION**	Your system administrator will use this option to define a user environment for your BINT session with BCOM
2	**BCOM**	Remote Data Management — use this option to start a Menu-Driven BINT session with BCOM. You must also supply the name of a BCOM-TSO environment.
T	**TUTORIAL**	A Help Tutorial on using the ISPF User Interface of BCOM
X	**EXIT**	Returns to TSO/ISPF

Navigating Through the User Interface

The ADMINISTRATION and TUTORIAL options are independent of MAIN TASK sessions and do not require a session to be established with BCOM in order to be used.

If you select the BCOM option, then the User Interface will attempt to establish a session with the Main Task, using one of the APPLID specified in the environment profile supplied on the option call.

Figure 4-2: Panel Navigation



Displaying the panel IDs

The User Interface supports special commands that control the panel displays. One of these is the PANELID command: to control whether the identifier of each panel will be displayed on the top leftmost corner of your screen, enter the following command on the COMMAND or OPTION line at the top of any screen:

```
PANELID  ON | OFF
```

PCCU and BUILD Macros

The MAXDATA operand of the PCCU macro determines the largest amount of data that may be sent from a VTAM application to the NCP.

You may define the MAXDATA operand of the PCCU macro with a value up to the largest possible. To be exact this value should be equal to BCOM's MAXBLK, plus the size of the SNA headers for transmission to an NCP: e.g. approximately 30 bytes of TH and RH.

If you want to use a BCOM MAXBLK of 4096, then your PCCU macro should set its MAXDATA to 4126. If you cannot change the NCP definition, then you can reduce the maximum size of the BCOM MAXBLK (e.g. 4096) by some 30 bytes - e.g. 4066.

(See *BCOM for MVS - Installation and Operations Guide* manual, *Restoring and Configuring the BCOM Components* Chapter, *Creating the Configuration File* Section, *Defining the LOCAL LOCATION* Sub-section, *FS62LCL Macro* item.)

Administration Functions

In order to create and maintain session environments or profiles for BCOM users, you, or the system administrator, will select the option ADMINISTRATION FUNCTIONS from the General Menu.

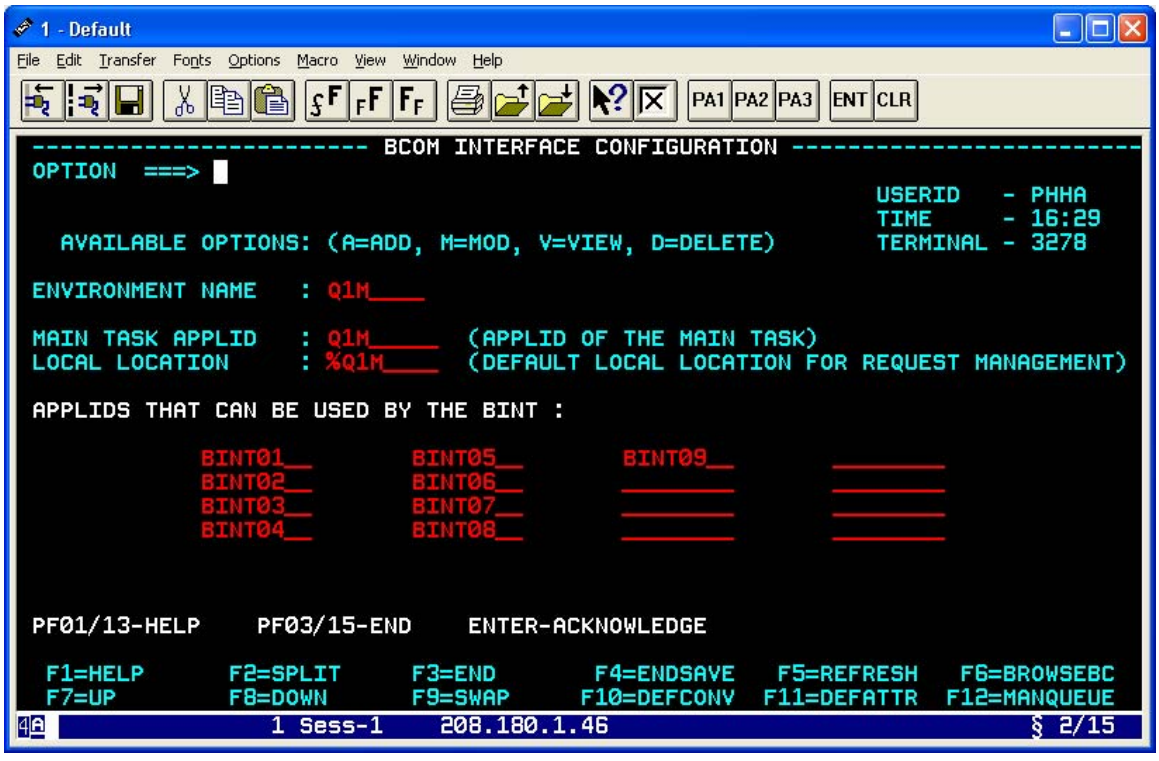
A BCOM control password is required to gain access to the INTERFACE CONFIGURATION panel. Please refer to *BCOM for MVS - Installation and Operations Guide,* section "ISPF-Administrative Functions" of "Configuring for the ISPF-Menu API".

User session environments

Some control information must be configured into the User Interface before a TSO user is allowed to initiate a session with BCOM.

This control information is called a 'session environment' and its characteristics are set by selecting the option ADMINISTRATION FUNCTIONS and replying to the password prompt.

Figure 4-3: User Session Environments



The Session Environment Profile

The Session Environment Profile is used to hide certain details of the User Interface and to pre-set some of the parameters needed to establish their BINT session.

BCOM is an SNA application: a BCOM Main Task Monitor must be accessed through a unique VTAM APPLID and even the MVS User Interface program will use APPC protocols to communicate with the Main Task Monitor. The Session Environment Profile defines the necessary VTAM APPLIDs to be used in establishing BINT sessions with a particular BCOM Main Task.

This is especially useful when running different copies of BCOM at the same installation.

Its content is described below.

ENVIRONMENT NAME	This name is unique per ISPF Table dataset and serves to store the characteristics being set by this function.
MAIN TASK APPLID	This is the VTAM APPLID of the BCOM Main Task.
LOCAL LOCATION	This is the default local location for the indicated MAIN TASK APPLID. This name must begin by a % sign and will automatically be inserted in your request definitions under the keyword LOCAL-LOCATION.
APPLID's TO USE	This is a list of all the VTAM APPLIDs that can be used by the User Interface in establishing communication with the Main Task of BCOM.

Note: *Do not forget to add to VTAM the APPLIDs defined in this environment profile - see the installation procedures in your* BCOM for MVS - Installation and Operations Giude*.*

Using the Session Environment Profile

There are different ways in which you can use a Session Environment Profile; for example:

- TO DISTINGUISH BETWEEN DIFFERENT BCOM ENVIRONMENTS - by pre-setting the name of a TEST or a PRODUCTION BCOM APPLID and local location name, your users don't have to know the WHERE's and HOW's in order to connect to the right BCOM program.

- TO CONTROL THE NUMBER OF CONCURRENT USERS - defining a short list of APPLIDs for the Interface will limit the number of concurrent sessions started with the User Interface.

Note that the ADMINISTRATION FUNCTIONS of the BINT are completely "stand-alone": they do not require any communication session with BCOM.

Administration Functions

The following Administration Functions can then be selected from the session environment panel:

A	**ADD**	To ADD a new environment configuration to your ISPF Table library. The environment name must be unique.
M	**MODIFY**	To MODIFY an existing environment configuration. The environment name must exist in your ISPF Table library.
D	**DELETE**	To DELETE an existing environment configuration. The environment name must exist in your ISPF Table library.
V	**VIEW**	To VIEW an existing environment configuration. The environment name must exist in your ISPF Table library.

BCOM Functions

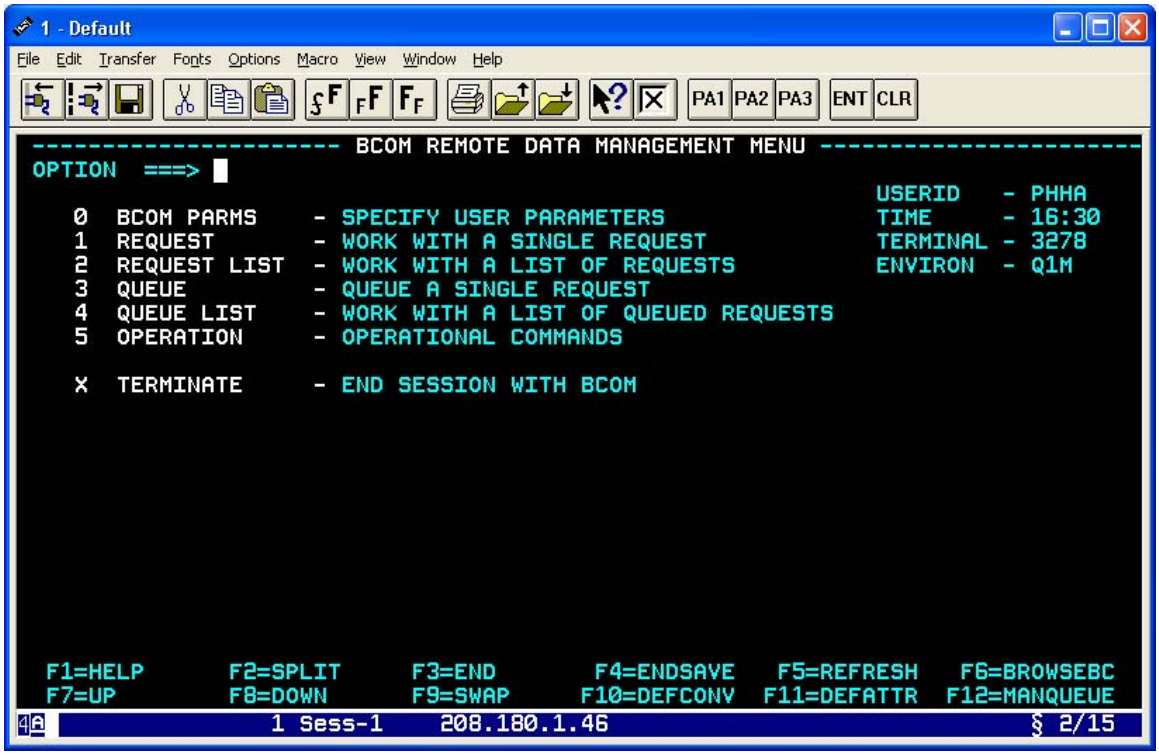
We will now describe the Remote Data Management functions of BCOM - or the BCOM Interface (BINT) proper.

The BINT Menu

To select the BINT option from the general menu, you must supply the name of a 'session environment', as defined through the ADMINISTRATION functions; the User Interface will use the control information contained in this profile to establish a BINT session with the appropriate BCOM Main Task.

This session will remain active until you exit from this panel (using PF3).

Figure 4-4: Remote Data Management Menu



The BINT Menu Functions

The BINT Menu Functions are organized into three categories: Request Management functions, Queue Management functions and Operational commands. The concepts behind these functions will be explained more thoroughly as the panels for each category are introduced.

A third option of the BINT menu screen allows you to pre-set some of the request definition keywords - or the default parameters of request definition.

0	**BCOM PARMS**	Use this option to set or change default values for certain keywords of request management
1	**REQUEST**	Use this option to work with a single request definition
2	**REQUEST LIST**	Use this option to work with a list of all the requests defined to BCOM
3	**QUEUE**	Use this option to QUEUE a single request definition
4	**QUEUE LIST**	Use this option to work with a list of all the requests that have been queued to BCOM
5	**OPERATION**	Use this option to work with the operational commands of BCOM
X	**TERMINATE**	Use this option to exit the menu.

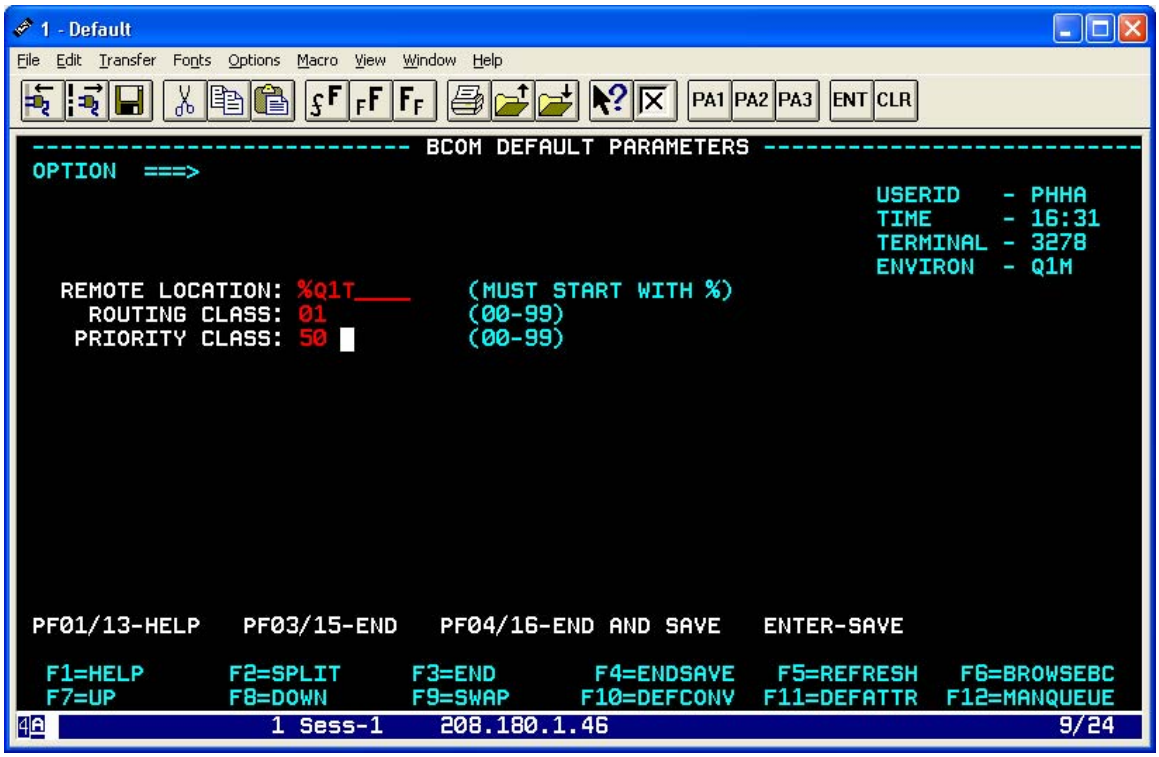
Default Request Management Parameters

With BCOM, all data transfer activities (with the exception of print distribution) take place on the basis of pre-defined requests - hence, the need for Request Management functions.

Request Management functions allow you to create BCOM requests using special parameters on data-entry panels. Entering these parameters can become laborious when you have a lot of similar requests to build.

Option 0 of the BINT menu allows you to pre-set the values of certain parameters: this, in turn, accelerates the request definition process and can minimize the possibility of data-entry errors.

Figure 4-5: Default Parameters



You can select the BCOM PARMS option anytime during your session with the User Interface: the option's panel will prompt you for the following default parameters of request management:

REMOTE LOCATION	Enter the name of a BCOM remote location — the name must begin with the symbol %.
ROUTING CLASS	Enter a valid routing class value for the configured BCOM program (00 to 99).
PRIORITY CLASS	Enter a valid priority class value with which BCOM will schedule your queued requests (00 to 99).

- If you press PF3/15, this will end the BCOM PARMS dialog without saving any changes
- if you type something and press PF4/16, then this will end the dialog while saving your default values
- pressing ENTER will also save the values specified but will not exit the dialog.

Request Management Functions

Concepts

The BCOM Request Definition

BCOM always takes its directives from a pre-defined script when transferring data between heterogeneous platforms. You can build such scripts using the Request Management functions of the User Interface - or the programmatic interface of BCOM.

In BCOM, scripts are called 'request definitions'. A request definition determines what needs to be transferred, where it will go, how it will get there and be processed, as well as when it should be sent.

The Request Management panels allow you to define a transfer request and to save it on a common repository file calls the 'Request File". Every BCOM location maintains its own request file. There are several advantages to having a central request file:

- you do not have to look for it, as BCOM manages it centrally;
- you can re-use any of the entries in the file, without having authored them (provided, of course, that your TSO LOGON user ID is authorized to access the resources referenced by them);
- you can chain any of those entries using the follow-on scheduling facility of BCOM;
- since the request name is exchanged between sending and receiving platforms, this gives you a comprehensive log of events and transfer activities on both sides;
- each request definition constitutes a record of things to do or things done: because requests can be re-used, this eliminates potential data entry errors and can also serve some auditing purposes.

The Display Formats

The panels of the Request Management function have been designed to support a dual mode of presentation:

- SINGLE ENTRY MODE - the single entry mode allows you to work with a single request at a time;
- LIST ENTRY MODE - the list entry mode allows you to scroll through a list of requests and select the one(s) that you want to work with.

If you choose to work with a list of requests, then you may select more than one item at a time: the user interface will process entries sequentially in the list and prompt you to continue, between each entry.

Working With A Single List

Anatomy of a Request

A BCOM request contains different types of information because several actions must be taken in order to properly schedule, transmit and complete a data transfer. If you look at the anatomy of a BCOM request, you will find the following categories of information:

REQUEST NAME
All BCOM requests must be identified and referenced by a name; this name begins with the symbol #.

TYPE OF DATA TRANSFER
For example: file distribution, backup/restore, remote job submission.

ROUTING INFORMATION
Including the names of the local and remote locations, the routing class to be used.

PROCESSING ATTIBUTES
Such as the priority class, check-pointing, compression, follow-on scheduling, etc.

LOCAL FILE INFORMATION
The name and allocation attributes of the file or job at the local location.

REMOTE FILE INFORMATION
The name and allocation attributes of the file or job at the remote location.

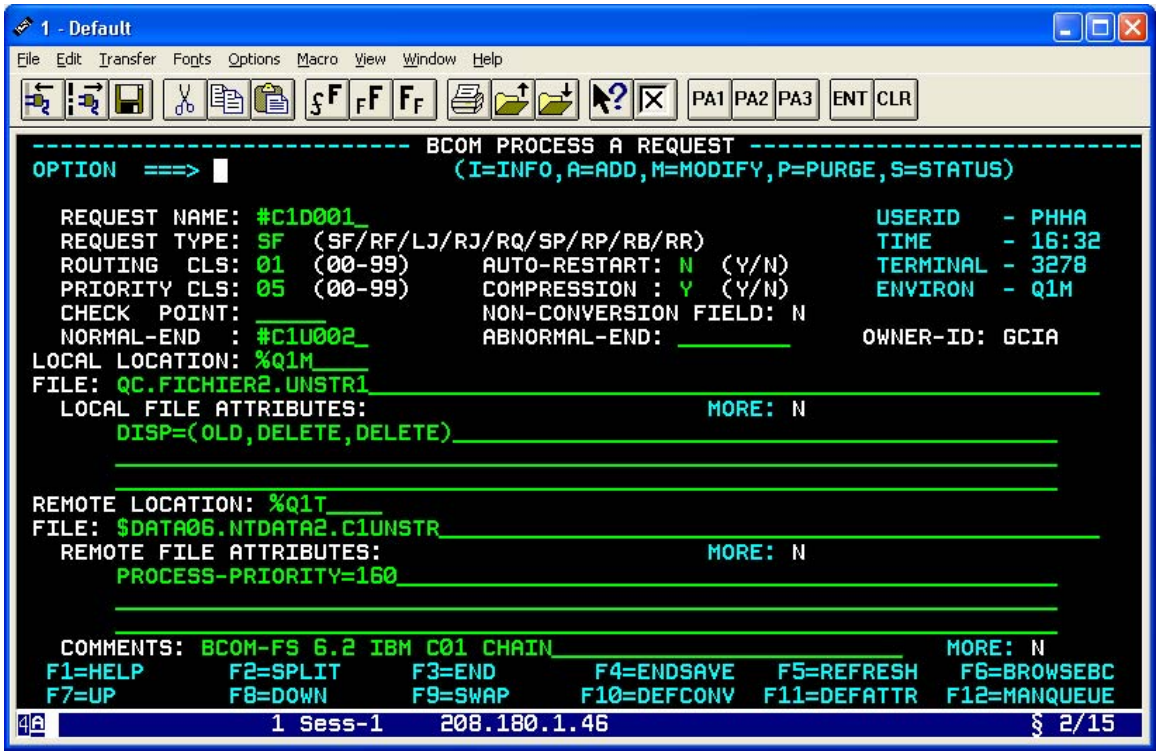
The panels of the User Interface are designed to let you specify, modify or purge those types of information using a minimum of data-entry or display panels.

The Request Definition Panels

Shown below is the first panel displayed by the User Interface when you choose to work with a single request definition: the panel tries to group together into a single display, all of the definition keywords that you are most likely to use when creating or modifying a request.

The input fields of this panel correspond to the KEYWORDS of the SET command, as supported by the command-line interface of BINT. The minimum number of fields that define a complete request will depend on the request type. For additional information on request building and the fields to supply to the BINT User Interface, please refer to the Command Reference appendix of your *BCOM User's Guide* - or to the Request Definition Keywords section of this document.

Figure 4-6: Request Definition Panel



Once you have entered all required fields, press ENTER to add the request. Remember that the request name must begin with the special character # and must be unique in the request file of your local-location.

The Request Definition Panel illustrated above normally suffices to build a simple request. If you have to supply additional information regarding non-conversion fields, local/remote attributes and/or comment lines, then other panels can be requested through the appropriate PF keys - as explained below.

Non-Conversion Fields

When the target or remote location of your data transfer is an ASCII machine, BCOM automatically performs character code translation for you. But if you must omit translation, then press PF10/22 to display the non-conversion fields' panel, and enter a list of the fields that should not be translated in your data records. BCOM assumes that the record layout is the same for all records in the file transferred.

Non-conversion fields are specified as one or more pairs of offsets and lengths. You can also enter the keyword ALL to bypass translation entirely. See *BCOM File Transfe*r chapter – *Data Conversion Utility* section for additional information.

When a request contains some non-conversion fields, a "Y" appears in the "MORE" field associated with non-conversion of the first request definition panel.

Non-Conversion fields operate on the principle that all the records in your data file have the same data layout; thus, by specifying a series of offsets and lengths, you can identify multiple 'areas' of the records to be exempt from character code translation.

Each 'area' of the record is defined by a number pair: "OFFSET,LENGTH".

If you want to bypass translation for the entire record, then either you specify a single number pair with an OFFSET of 0 and a length equal to the entire record size - or you type the special keyword "ALL" into the first input field.

When specifying multiple FIELDS, be aware that the results are cumulative. A warning will be issued at file transfer time if an offset,length specification exceeds the record size of the file.

File Attributes

The principal panel has room for only three local-file attributes and three remote-file attributes; if you need to define more attributes, press PF11/23 and an overflow panel will appear in which you will be able to enter the additional attributes.

When a request contains more than three attributes of either kind, the corresponding "MORE" field in the first request definition panel is set to contain a "Y" value: you can then press PF11/23 to view the additional attributes.

Comment Lines

The first panel has room for a single comment line; if you need to define additional comment lines, then press PF11/23 and an overflow panel will appear to let you enter up to 4 additional comment lines describing your request.

When a request contains more than one line of comments, the corresponding "MORE" field in the first request definition panel is set to contain a "Y" value-you can then press PF11/23 to view the additional comment lines.

Request Definition Functions

The following functions can be selected from the Request Definition panel:

I	**INFO**	Use this option to obtain information about an existing request definition.
A	**ADD**	Use this option to ADD your completed request definition to the request file of BCOM. The user interface will make sure that the request name supplied is unique.
M	**MODIFY**	After using the I (INFO) command for an existing request, you can change the contents of a given request and use the M (MODIFY) option in order to record those changes.
P	**PURGE**	You can use this option to purge the indicated request from the BCOM request file.
S	**STATUS**	Used in conjunction with the name of a request, this option effectively transfers you to the Queue management functions and returns a status of the queued request.

Notes on the above commands:

INFO *You can modify the contents of a request using the result of an INFO command; this technique allows you to create a new request using an existing one as a model.*

PURGE *One of the following conditions must be true before a request can be purged: (1) you are the request OWNER, (2) the Owner ID is all blanks, or (3) you have logged on under a MASTER OPERATOR ID.*

Cross-Over to Queue Management

If you happen to know the exact name of a request to be queued - perhaps it is the one that you have just created in the request definition panel - then you do not have to exit all the way back to the menu panel and select the Queue management functions in order to schedule the request: the command line area of request management screens supports the following Queue management short-cuts:

PFR/24 displays the queue management screen and carries over any applicable request data - allows you to Queue, Queue-wait, set schedule data and override some request attributes.

Remember that a waited queue (QWAIT) will lock your keyboard until the request completes; at that time, the User Interface will unlock your keyboard and display the request completion status on your screen.

The Request Definition Keywords

The following fields of the Request Management panels are used to define a BCOM request:

Request name (required field):
the name of a request definition must begin with the symbol #.

Request Type (required field):
defines the type of data transfer operation. It can take one of the following values:

SF (SEND-FILE)
to send a local file to a remote location.

RF (RECEIVE-FILE)
to receive a remote file into a local file.

LJ (LOCAL-JOB)
to execute a JOB at the MVS local location the keyword LOCAL-FILE or REMOTE-FILE determines where the job data stream resides.

RJ (REMOTE-JOB)
to execute a JOB at the remote location the keyword LOCAL-FILE or REMOTE-FILE determines where the job data stream resides.

RQ (REMOTE-QUEUE)
use this type to queue a request at the remote location; the request name is provided by the REMOTE-FILE keyword

RB (REMOTE-BACKUP)
use this type to trigger a backup request whose definition is in a remote BCOM platform Request File.

RS (REMOTE-RESTORE)
use this type to trigger a restore request whose definition is in a remote BCOM platform Request File.

*** SP (SEND-PDS)**
specifies a PDS to PDS file transfer from the local location to the remote location.

*** RP (RECEIVE-PDS)**
specifies a PDS to PDS file transfer from the remote location to the local location.

* Valid for transfers between MVS's only.

The Routing Information

ROUTING CLASS
Required field. Valued 00-99; must correspond to one of the routing class values defined under the specified remote-location in the BCOM configuration file.

LOC-LOCATION
Required field. Name of the BCOM LOCAL-LOCATION (i.e. your current BINT session); name must appear in the FS62LCL macro of the configuration file for that MVS node. This field will be pre-set with the value that was defined in your environment profile. All BCOM location names must begin with the symbol %.

REMOTE-LOCATION
Required field. Name of the BCOM REMOTE-LOCATION; name must appear in one of the FS62RMT macros of the configuration file of the local MVS node. This field will be pre-set with the value that was defined through your BCOM parms option (0). All BCOM location names begin with the symbol %.

The Processing Attributes

PRIORITY-CLASS
Required field. Valued 00-99 in descending order of priority (although in FIFO order within the class): a class of 00 will be processed first, a class value of 99 will be processed last. Default value is 05.

CHECKPOINT
Optional field. Enter a checkpoint interval corresponding to a number of records in the file. Try making it a multiple of the BCOM configured window count.

AUTO-RESTART

Optional field. Cause BCOM to automatically re-initiate the request if aborted due to communication line or software error.

COMPRESSION

Optional field. If you specify "Y" then make sure that the routing class in the BCOM configuration file is also configured to support compression.

NORMAL-END

Optional field. Name of the next (local) request to be scheduled by BCOM when this request completes NORMALLY.

ABNORMAL-END

Optional field. Name of the next (local) request to be scheduled by BCOM when this request completes ABNORMALLY.

NON-CONVERSION FIELD

Optional field. For EBCDIC-ASCII character set translation. If you don't want BCOM to translate certain fields of your file records, then press PF10/22 and supply the requested information on the panel displayed. If the request you are viewing contains non-conversion fields, then a "Y" will appear in this field.

The Local File Information

OWNER-ID

- AUTO-ENTERED FIELD: Shows the LOGON USER ID that created and now owns this request. Request is not owned if OWNER-ID field is blank.

- To force request ownership, please see the procedure described in the BCOM Version 4.0 Migration Guide.

LOC FILE

Name of the local file. Requirement depends on request type:

- Required for SF, RF, SP and RP;
- Optional with LJ and RJ;
- Not used with RQ.

LOC-FILE ATRIBUTES

Optional field. You can enter up to 3 local-file attributes on this panel; press PF11/23 to enter more. Local file attributes will refer to file allocation attributes (e.g. JCL).

For the list of local file attributes and their parameters, please refer to the LOCAL-ATTRnn identifier the SET command in *Appendix A*.

The Remote File Information

REMOTE-FILE

Name of the Remote file. Requirement depends on request type:

- Required for SF, RF, SP and RP;
- Optional with LJ and RJ.
- Required with RQ: it defines name of the request to be queued at the remote location (must exist in the Request file at the remote-location).

REMOTE-FILE ATTRIBUTES

Optional field. You can enter up to 3 remote-file attributes on this panel; press PF11/23 to enter more. Remote file attributes depend on the type of allocation functions supported by the remote platform and the type of BCOM request that you are creating.

For remote file attributes, please refer to the LOCAL-ATTRnn parameter for NSK Guardian in the *BCOM for NSK Guardian – User's Guide*; Or the FILEATTR parameter in the *BCOM for Windows NT - User's Guides*; the LOCAL-ATTRnn identifies the SET command in *Appendix A* at the end of this guide.

IMPORTANT NOTE:

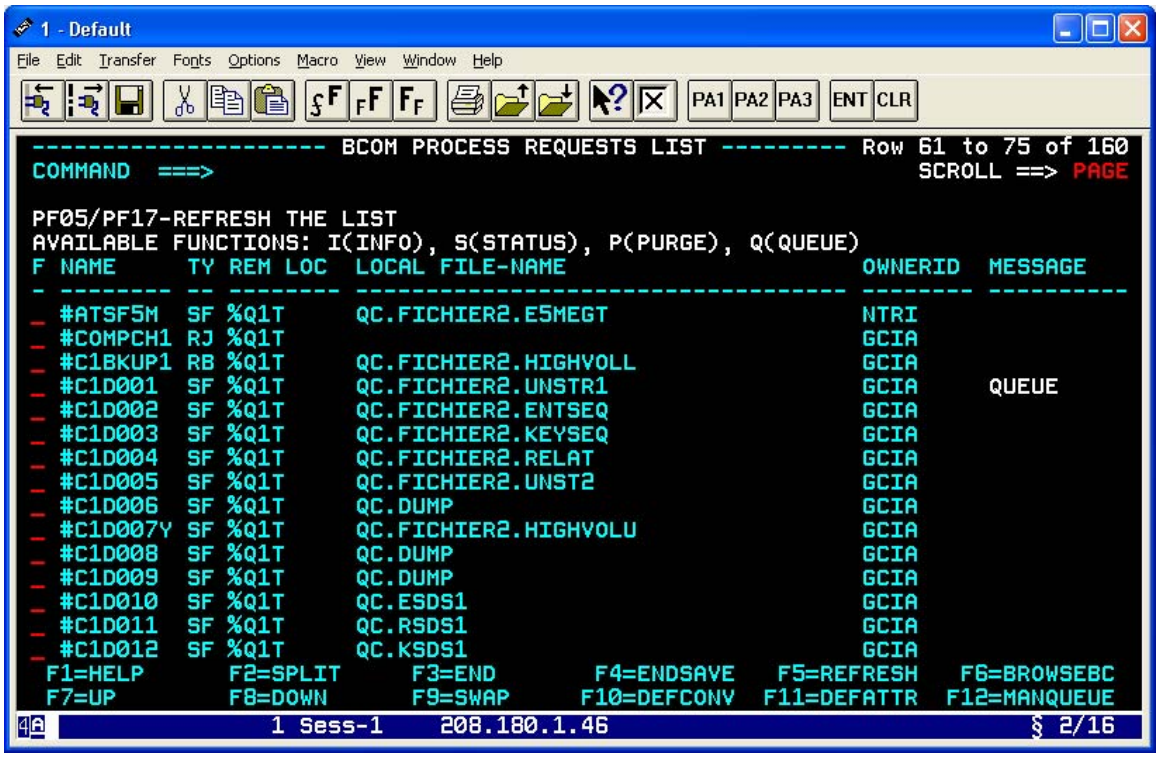
Only ONE attribute is allowed per display line on the ISPF full screen of the BINT.

Work With Request List

A different way of working with the Request Management functions of BCOM is to look at all the entries in the request file of BCOM and select the ones that you want to work with.

The list entry mode of Request Management (Option 2) allows you to do that; you can scroll through the entire list of request definitions and enter the appropriate function code next to each entry to be processed.

Figure 4-7: Work with Request List



Fields in the Scroll List

The scroll list provides the following information about each request:

REQUEST NAME
The name of the request definition.

REQUEST TYPE
The type of request such as SF (Send-File), RF (Receive File), LJ (Local-Job), RJ (Remote-Job), RQ (Remote-Queue), SP (Send PDS - MVS to MVS only) or RP (Receive PDS - MVS to MVS only).

REM LOC
The name of the REMOTE LOCATION referenced by this request.

LOCAL FILE-NAME
The name of the LOCAL-FILE referenced by this request.

OWNER-ID
The LOGON USER-ID that owns this request (and can purge it).

MESSAGE
A status of the last operation performed against this entry.

Functions Supported

In front of each display line, the first input field accepts a FUNCTION code which lets you take some action against the corresponding request:

I	**INFO**	Use this option to obtain information about an existing request definition. You can then modify its contents.
P	**PURGE**	You can use this option to purge the indicated request from the BCOM request file—provided you are the request OWNER ID, the latter is all blanks, or you have logged on under the MASTER OPERATOR id.
S	**STATUS**	Used in conjunction with the name of a request, this option effectively transfers you to the Queue management functions and returns a status of the queued request.
Q	**QUEUE**	To queue asynchronously a request having the current request name. The User Interface will bring you to the "PROCESS A QUEUED/DEQUEUED REQUEST" panel where you can enter the queuing parameters (if any).

Creating a New Request

You can select one of the entries in the list to be your MODEL for a new request; simply select the desired request from the list using the INFO function, and press enter. When in the "PROCESS A REQUEST" panel, make the necessary changes to that request and then enter an ADD command to create the new request.

The Message Field

Once the selected entry has been processed, the function code is removed and a completion message appears in the MESSAGE field to remind you of the action taken.

For example, purging a request will generate the following message: "*PURGED" for that particular entry. Since the actual entry has been removed from the repository file, you can refresh the display using PF05/17: e.g. now causing purged entries to disappear.

Making Multiple Selections

The User interface can handle multiple selections on the same screen. Here is how you use this feature:

- enter a processing option code into the "F" (or FUNCTION) field of two or more request entry lines;
- press ENTER to acknowledge.

The User Interface will process the entry selections as follows:

- the first entry in the current screen list with a non-blank function code will be serviced;
- the function code will be removed when the entry has been processed;
- a completion status will be shown in the message field;
- the User Interface will prompt you to continue before processing the next non-blank entry in the list.

The Request Status

Note that BCOM preserves the status of the last queue or request management operation performed on a defined request and makes that information available to you via the STATUS command (see Appendix A for more information).

For example, if the last operation performed on a given request was a QUEUE delete command, then the status information screen will indicate that the entry was last deleted.

Queue Management Functions

Concepts

The BCOM Queue

Once a request has been defined and stored into the Request file using the Request Management functions, it is ready to be queued for execution.

Request queuing overcomes the possibility that the state of network connections between heterogeneous platforms may fluctuate. Queuing also simplifies error recovery and certain operational activities such as time-dependent scheduling. There are two forms of request queuing:

- synchronous queueing returns control when the transfer has completed and is achieved with the QWAIT command;
- asynchronous queueing returns control as soon as the request is placed in the queue for transfer and is achieved by using the QUEUE command.

When a request is queued, BCOM copies the request definition to a temporary staging area called the queue file: every BCOM location has a queue file which it uses to keep track of the progress of transfer requests.

BCOM requests are dequeued and scheduled according to optional time-scheduling information, and are processed in FIFO order within the priority-class that was assigned when the request was first created. If you want, you can override this priority-class by entering an additional parameter on the Queue command. Time-based requests will stay into the queue file until their scheduling time has arrived-or a QDEL command is issued for them.

A BCOM request will transfer data between the local-location and a designated remote location; to do so, BCOM needs to queue the request to a routing class that services such a remote-location. It is your responsibility, as a user, to assign a valid routing class to your request definition: please consult your BCOM configuration file for more information.

If your BCOM configuration file defines multiple routing classes for the same remote-location, then you can override this value by supplying an optional routing-class parameter at Queue time: you would do this to take advantage of alternate paths between the same two BCOM nodes.

The actual data transfer will not begin until the connection to the remote location has been established (connections are normally established and maintained automatically by BCOM) and a free local requestor process (or PROCID) has been found to service that request.

Once a request has been queued, other functions of Queue management can be used to manage the contents of the Queue file: queued requests can be displayed, held, released, aborted or manually re-started (if using check-pointing).

The Last Transfer Statistics

When a transfer completes, the queue record is deleted but the transfer statistics accumulated during processing are saved back into the request definition entry, inside the request file. You can review that information which includes the start and end time, number of records, completion status etc. with a STATUS command for that request entry.

The Display Formats

The panels of the Queue Management function have been designed to support a dual mode of presentation:

- SINGLE ENTRY MODE - the single entry mode allows you to queue a single request at a time
- LIST ENTRY MODE - the list entry mode allows you to scroll through a list of queued requests and select the one(s) that you want to work with.

If you choose to work with a list of queued requests, then you may select more than one item at a time: the user interface will process entries sequentially in the list and may prompt you to continue between each entry, or process each entry without a prompt. (For example, several STATUS commands).

Queueing A Single Request

Queuing Commands

Various commands allow you to queue or re-queue a request, or modify the execution of an already queued request:

Q Entering this function code will queue your request for execution and then allow you to enter other commands or return to another screen.

W Entering this function code will queue your request for execution and wait for the transfer to complete before returning control to your screen. You are not allowed to specify a scheduling date, time or repetition parameter on a W command.

P Use this command to alter the priority-class of the indicated request; tab to the priority field and enter a valid priority value before pressing the enter key.

D Use this command to delete the indicated request. The request must not be executing to be deleted.

A Use this command to abort the indicated request. The request must be “EXECUTING” in order to be aborted.

H Use this command to hold a queued request that has not yet executed.

L Use this command to release a queued request that was held for execution.

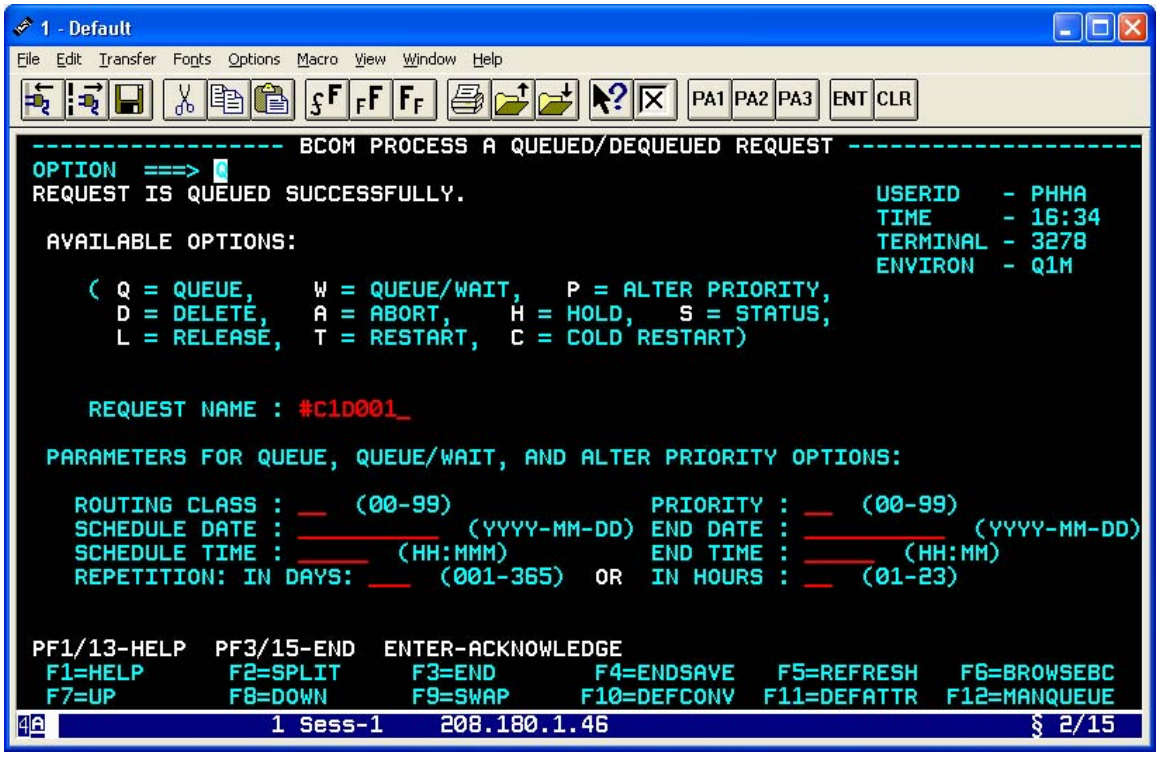
T Use this command to restart a queued request that ended abnormally and is in state "HELD FOR RESTART". This occurs when the request makes use of the checkpoint restart facility. BCOM will attempt to restart the transfer from the last successful checkpoint.

C Use this command instead of the 'T' command, if you choose to ignore or clear the checkpoints completely when restarting a check-pointed request HELD for RESTART. This will have the effect of starting the transfer as if the request were queued for the first time.

S Entering a status command for a given request will present you with a status of the queued request. Please see the STATUS option in the OPERATIONAL COMMANDS MENU and the STATUS command in *Appendix A*, for more information on a request status.

REQUEST panel of the User Interface:

Figure 4-8: Queue a Request



Queueing Parameters

Various options are available from this screen in order to queue or re-queue a request, or to modify the execution of an already queued request:

ROUTING CLASS

Enter a class value into this field if you wish to override the routing class already defined for the request to be queued.

PRIORITY CLASS

Enter a class value into this field if you wish to specify a new priority class for an already queued request.

SCHEDULE DATE and TIME

These two optional fields are used to specify a start time for the execution of the request. BCOM will maintain the request on its queue file until the specified schedule data and time have arrived.

REPETITION (in DAYS or HOURS)

If you have already specified a schedule date and time, then you can also specify a repetition factor in days or hours. BCOM will then automatically re-schedule your request every so many days or so many hours, depending on the option selected.

For additional information on these parameters, please refer to the instructions on creating a request, in the Request Definition Keywords section of this manual.

Considerations for T-Restart

Before restarting a check-pointed request with the 'T' command, make sure that the request definition was built in such a way as to be able to re-allocate the target file at the remote system (e.g. DISP=SHR for MVS).

Considerations for Delete

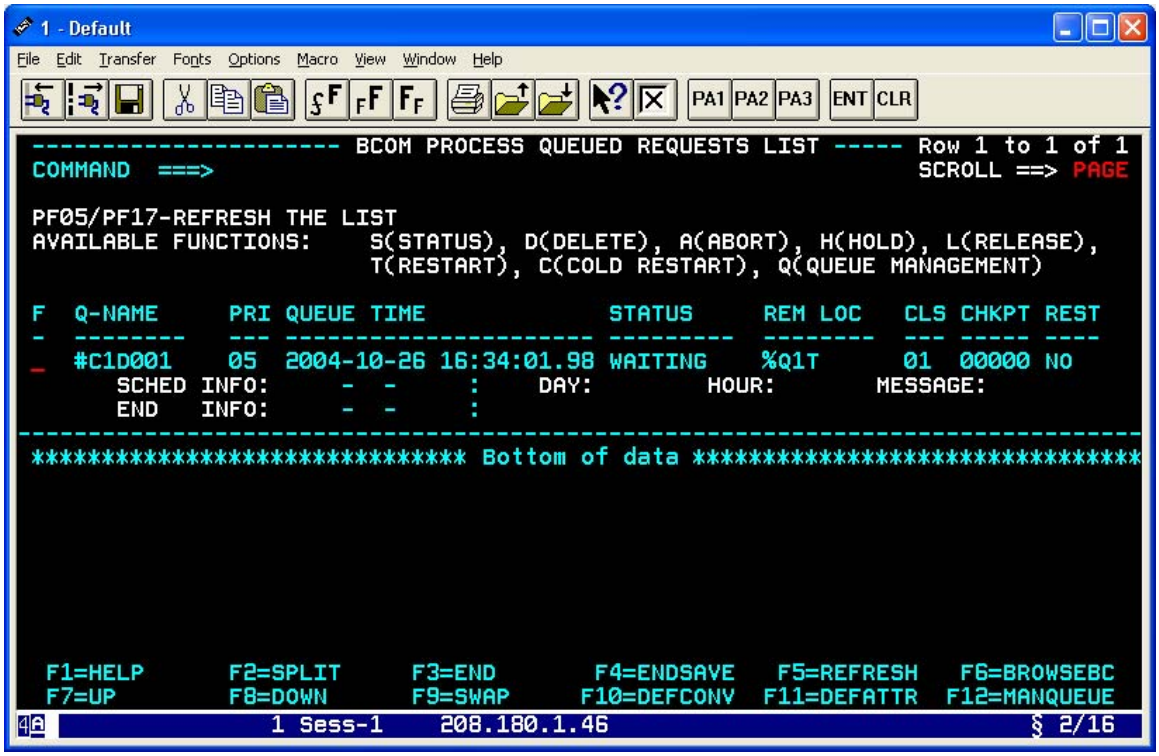
When a DELETE command is entered, the entry remains in the current list of the User Interface but its MESSAGE field is modified to indicate "DELETE OK". If you want to remove the deleted entry from the scroll list, then simply REFRESH the screen using PF05/17.

Queueing Requests From A List

A different way of working with the Queue Management functions of BCOM is to look at all the entries in the queue file and select only the ones that you want to work with.

The list entry mode of Queue Management allows you to do just that, by scrolling through the entire list of queued request and entering the appropriate function codes next to each entry:

Figure 4-9: Queue from a List of Requests



Multiple selections can be made from that screen: they will be processed in the same way as in the WORK WITH A LIST OF REQUEST screen described earlier in this section.

Fields in the Scroll List

The information provided on each request is displayed on two lines of information within a scroll list. The first line of information contains the following fields:

Q-name The name of the queued request.

PRI The priority-class with which the request is currently queued.

QUEUE TIME The date and time at which the request was initially queued.

STATUS The status of the current request.

REM LOC Name of the BCOM REMOTE-LOCATION referenced by this request

CLS ROUTING-class to which the request is currently queued.

CHKPT Current or last checkpoint value for a request that uses checkpoints.

REST Indicates (with YES | NO) whether a request is currently in restart.

The second line of the information contains optional scheduling fields and the MESSAGE indicator:

SCHED INFO Shows the scheduling Date and Time assigned to this request.

DATE This field is filled-in if a daily repetition factor was also added to the scheduling information when the request was queued.

HOUR This field is filled-in if an hourly repetition factor was also added to the scheduling information when the request was queued.

MESSAGE A status of the last operation performed against this entry.

Functions Supported

In front of each display line, the first input field accepts a FUNCTION code which lets you take some action against the corresponding request:

S	STATUS
H	HOLD
D	DELETE
L	RELEASE
A	ABORT
T	RESTART
C	COLD RESTART
Q	QUEUE MANAGEMENT

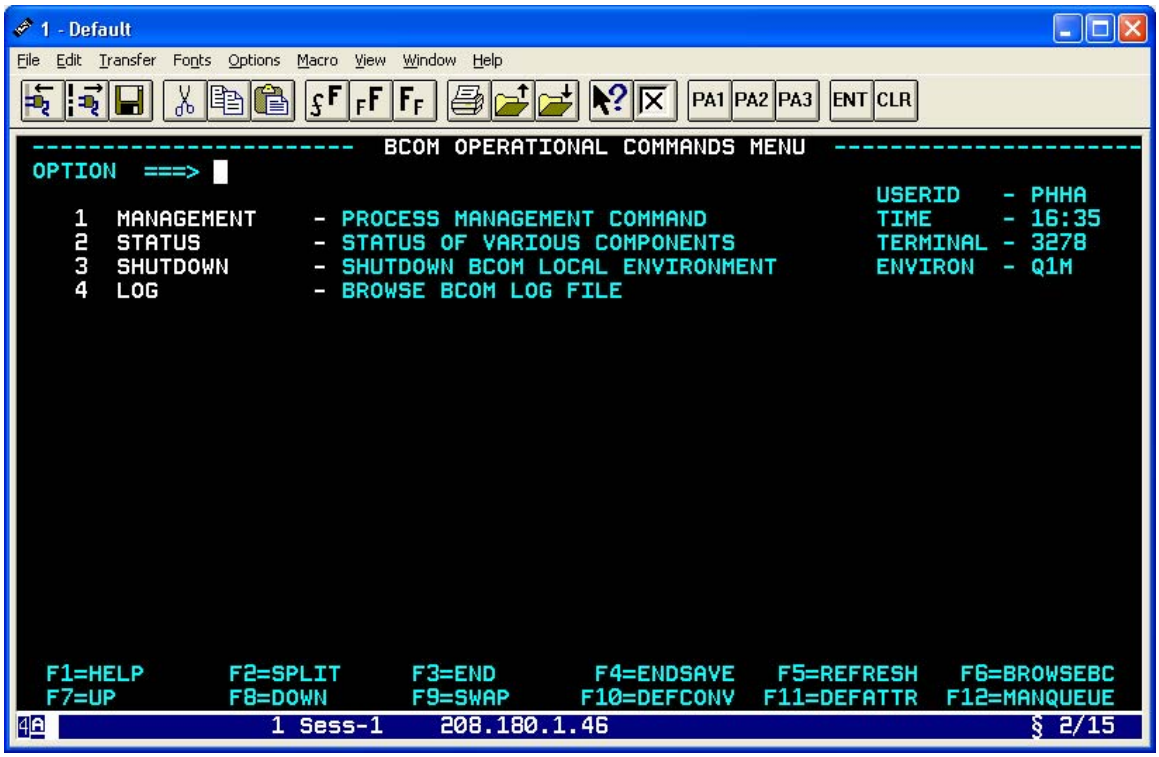
For more information on these options, please refer to the section on Queuing commands described earlier in this chapter.

Operational Commands

Organization of Operational Commands

To facilitate the use of BCOM operational commands, the latter have been organized under categories, as illustrated in the table.

Figure 4-10: Operational Commands Menu

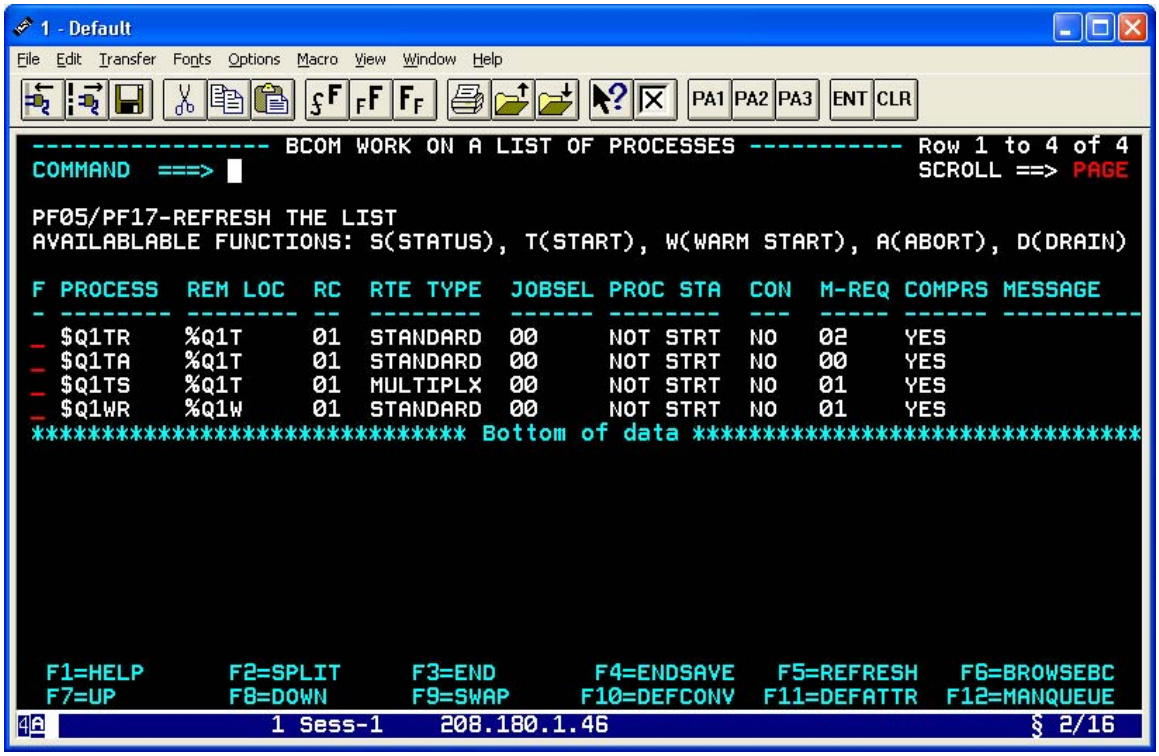


1	**MANAGEMENT**	Use this selection to display, control, start, stop the requestor/server processes of your local-location and thus maintain your connections to other BCOM platforms.
2	**STATUS**	This selection provides you with the means of finding the status of all critical components of BCOM.
3	**SHUTDOWN**	Please use MVS Command CANCEL <BCOM-jobname>.
4	**LOG**	This selection is used to browse the log of messages generated by your BCOM local-location.

Process Management Commands

The WORK ON A LIST OF PROCESSES screen displays a scroll list of all the BCOM remote-locations that are defined in your BCOM configuration file, along with the status of the requestor PROCIDs that have been assigned to each of those locations.

Figure 4-11: Process Management Commands



Command option codes can be entered into the Function ('F') field of any process entry to let you better manage the network connections between your BCOM local-location and the BCOM configured remote-locations. These will be explained in the next paragraphs.

Fields in the Scroll List

The information provided on each request is displayed on two lines of information within a scroll list. The first line of information contains the following fields:

F
This is the function field into which you type one of the available function codes.

PROCESS
The name of the process or PROCID as it appears in the configuration file of BCOM, under the indicated remote-location.

REM LOC
The name of the BCOM remote-location that is serviced by this PROCID.

RC
The routing class under which this PROCID operates when connecting to the indicated remote-location.

RTE TYPE
The type of routing class which this PROCID implements when communicating with the remote-location. Possible values include: STANDARD and MULTIPLX.

JOBSEL
This counter indicates the number of requests currently started and still awaiting for allocation of the target dataset to complete at the remote location.

PROC STA
This field indicates the current status of the process in BCOM.

CON
This indicator shows whether the PROCID has successfully connected to the remote-location partner.

M-REQ
The MAX-REQUEST is a value specified in the BCOM configuration file that controls the maximum number of transfer requests that this PROCID will service at a time.

COMPRS
This indicator shows whether the indicated PROCID is enabled to service transfer requests that use DATA COMPRESSION.

MESSAGE
A status of the last operation performed against this entry.

Functions Supported

For each entry in the scroll list, one of the following sub-commands can be entered into the Function (F) field:

S	**DETAILED STATUS**	This sub-command will provide you with a detailed status of the particular requestor process and of each of its LUs
T	**START PROCID**	Use this command to start the PROCID. This should eventually connect your BCOM local-location to the remote-location serviced by this PROCID.
W	**WARM START**	Use this command after a failure of the indicated PROCID to connect to the remote location; this could be due to a lost LU condition or to LUs of the PROCID that were never started.
A	**ABORT PROCID**	Use this command to terminate a PROCID: the process will ABEND and any request currently executing under that process will be terminated.
D	**DRAIN PROCID**	Use this command to stop a PROCID: the process will stop after requests currently executing under that process have all terminated. Meanwhile, no new requests will be accepted by this PROCID.

Handling Multiple Selections

As with other scroll lists, you can enter function codes for more than one entry on the screen: the User Interface will simply prompt you to continue between entries.

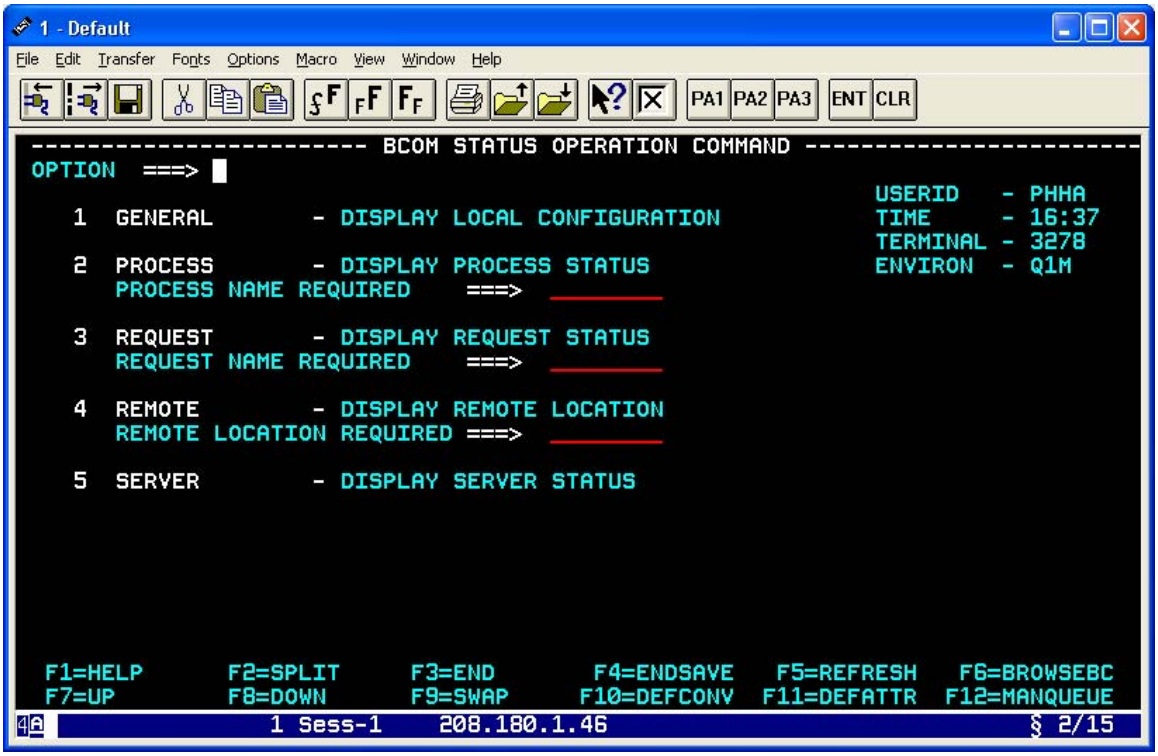
Considerations for Starting PROCIDs

After a START command, the message field will read "START OK". To follow the progress of the START and ascertain that the connection was made successfully, you should follow the START option with a REFRESH command (PF05/17) and monitor the PROCESS STATUS field.

General Status Commands

The STATUS OPERATION COMMAND screen shows a menu of STATUS commands to various types of BCOM components.

Figure 4-12: Status Commands



Options Available

The following options are available from the STATUS OPERATION COMMAND screen:

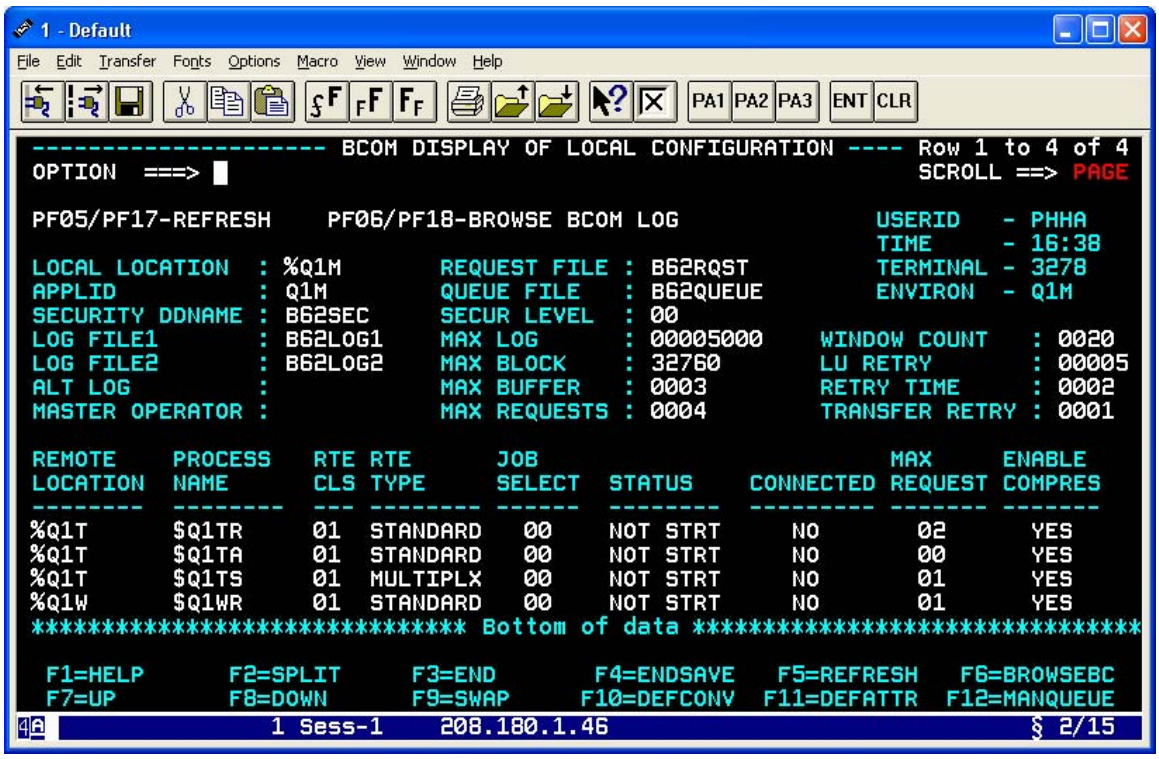
	ITEM	DESCRIPTION
1	**GENERAL**	Display the configuration information for the BCOM local-location.
2	**PROCESS**	Display the status of a specific BCOM requestor process (or PROCID).
3	**REQUEST**	Display the status of a request.
4	**REMOTE**	Display the status of a specific remote location.
5	**SERVER**	Display the status of the server processes (or PROCID) at the BCOM local-location.

Status of the Local-Location and Configuration

The purpose of this function is to provide an on-line feedback of the BCOM configuration settings currently in effect in the running task of BCOM

This feedback is especially useful after you make a change to certain parameters of BCOM operating environment and wish to confirm that the change has taken effect.

Figure 4-13: Local-Location Status



The display also summarizes, at a glance, the state of all network connections established, pending or stopped between this local location and all configured BCOM remote locations; use the standard scroll keys of TSO (PF07/PF08) to review that list.

The following field items echo most of the important configuration attributes of the BCOM local location macro (FS62LCL), as appearing in the BCOM configuration file in effect:

LOCAL LOCATION

The name of the BCOM local location with which your User Interface is in session.

APPLID

The name of the VTAM APPLID used by this BCOM local location to communicate with BINTs (like this User Interface) and BCOM remote locations.

REQUEST File
The MVS JCL DDname used by the local location to access the BCOM request file.

QUEUE FILE
The MVS JCL DDname used by the local location to access the BCOM queue file.

Log File1 , File2
The MVS JCL DDname used by the local location to access the BCOM dual log files.

Master Operator
The MVS USERID acting as the BCOM master operator; all BINT users are master operators if omitted (e.g. blank).

Security DDname
The MVS JCL DDname used by the local location to access the BCOM security file.

Security Level
The security level of the local location.

Max Log
The maximum number of records to be logged per log file.

Max Block
The maximum size of a transfer buffer between this local location and other remote locations; actual value is negotiated on connection.

Max Buffer
The maximum number of buffers to be allocated per request.

Max Requests
The maximum number of concurrent requests supported.

Window Count
The default window count for this location, indicating the maximum number of buffers sent before requesting an acknowledgement from the partner location.

LU Retry
The maximum number of times that BCOM will retry an LU or EP session in error.

Retry Time
The interval in minutes between retries.

Transfer Retry
The maximum number of times that BCOM will retry sending a file using a request that defines AUTO-RESTART (Future use).

Status of a Process (or PROCID)

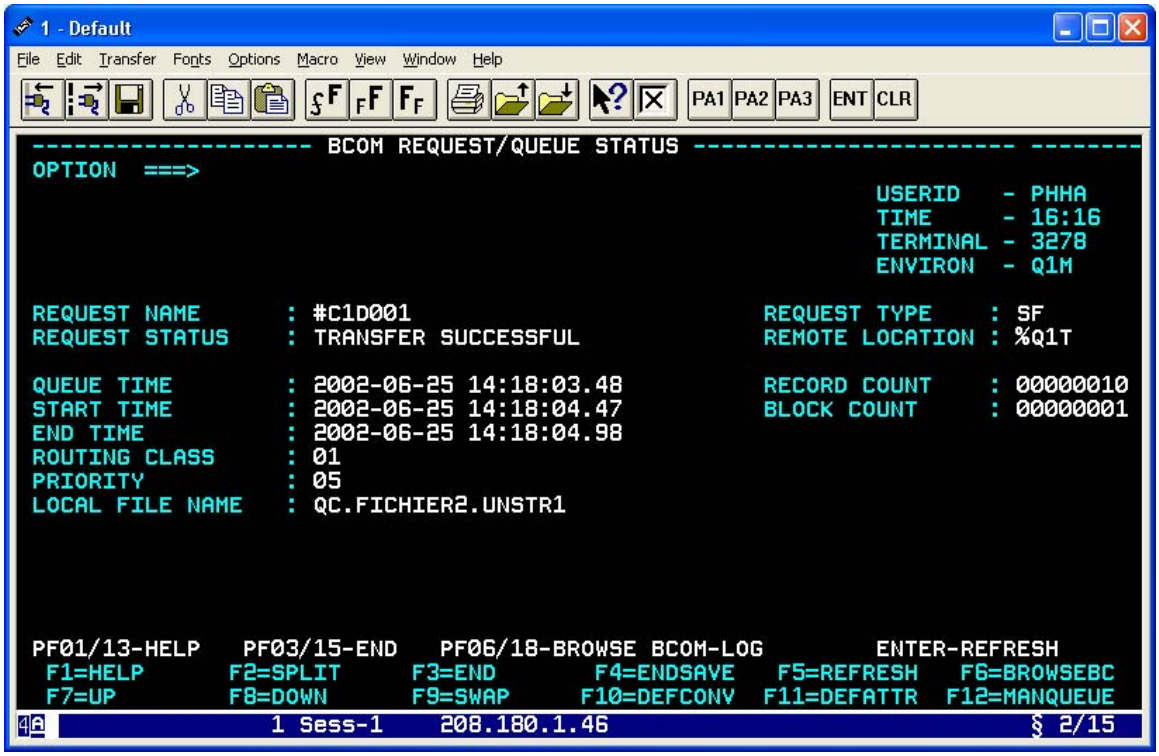
This display basically presents the same kind of status information as is displayed by a Process Management status command (from the Operation Commands Menu) except that the output is limited to a specific process (or PROCID).

Please refer to Process Management Commands in the *Operational Commands* section of this guide for more information on the display fields.

Status of a Request

This command displays status information concerning a request. This is the same display which is returned in response to a Status command issued from the Queue Management section of the User Interface, or by a Status function code entered from the Option line of other screens.

Figure 4-14: Request/Queue Status



These server functions are displayed using the following field items:

REQUEST NAME
The name of the request at this local location and whose informational status is now being displayed.

REQUEST TYPE
The type of transfer (e.g. SEND-FILE, REMOTE-JOB, etc).

REQUEST STATUS
Informational message indicating the progress of the transfer (e.g. waiting, executing, completed) or a completion status (e.g. transfer successful or some error condition).

REMOTE LOCATION
The name of the BCOM remote location involved in this transfer.

QUEUE TIME
The date-time at which the queue command was entered to queue this request.

START TIME
The date-time at which the request started executing.

END TIME
The date-time at which the request completed.

ROUTING CLASS
The routing-class to which the request was queued in order to communicate with the indicated remote location.

PRIORITY
The priority at which the request is queued.

LOCAL FILENAME
The name of the file referenced by this request at the local location.

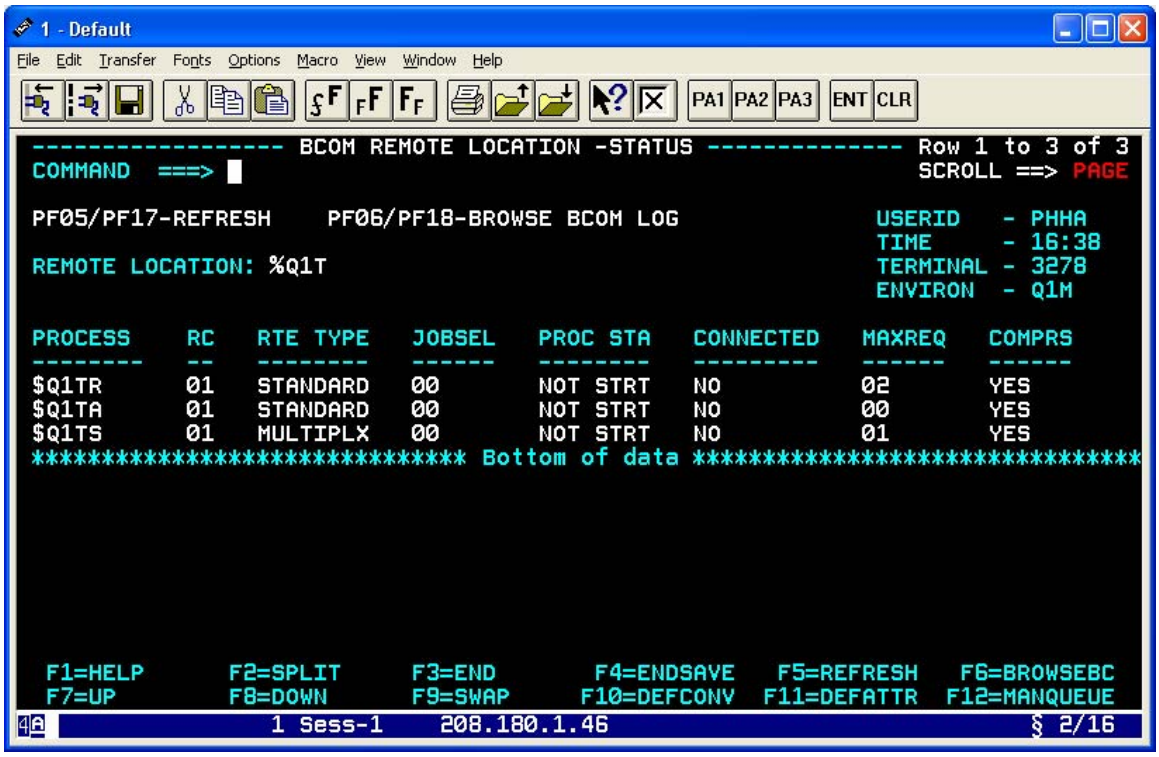
RECORD COUNT
The current or final number of records transferred.

BLOCK
The current or final number of blocks of records (SNA I/Os) transferred.

Status of a Remote Location

The display of a BCOM remote-location is a subset of the PROCESS MANAGEMENT display: it restricts the status display to the PROCIDs associated with a particular remote location.

Figure 4-15: Remote-Location Status



For a description of the fields please refer to the discussion of Process Management Commands, earlier in this sub-section.

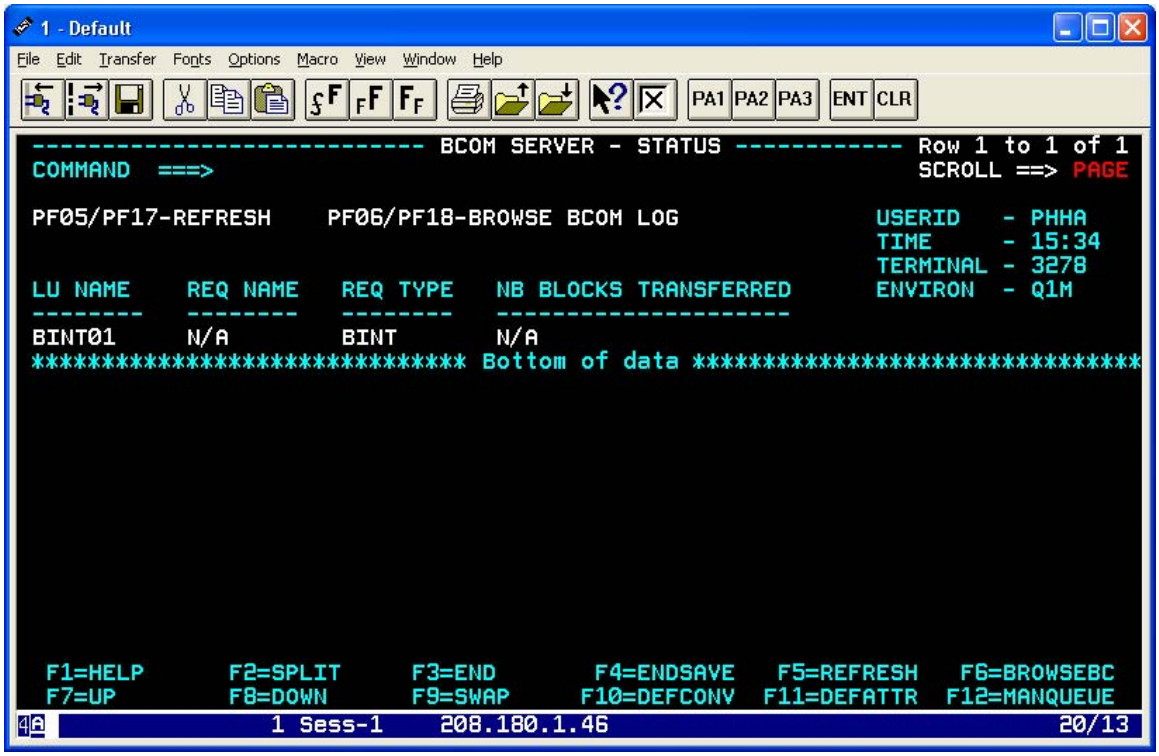
An alternative way of listing all configured remote locations is to select option 1 (GENERAL STATUS) from the operational commands menu and then review the configuration information for each of the configured BCOM remote locations.

Status of the BCOM Server Processes

Within the BCOM program for MVS, the SERVER processes come in two flavors:

- the server function that services transfer requests originating from BCOM remote locations
- the server function that services BINT requests originating from instances of the User Interfaces.

Figure 4-16: Server Status



These server functions are displayed using the following field items:

LU NAME
Name of the SNA LU (or BCOM EndPoint) servicing an incoming transfer request or name of the SNA LU servicing a BINT session with BCOM.

REQ NAME
The name of the BCOM request currently being transferred.

REQ TYPE
Either the type of request being processed (e.g. SF) or the literal 'BINT' indicating that this is a User Interface session.

NB BLOCKS TRANSFERRED
Applicable only when a transfer request being serviced.

The Shutdown Command

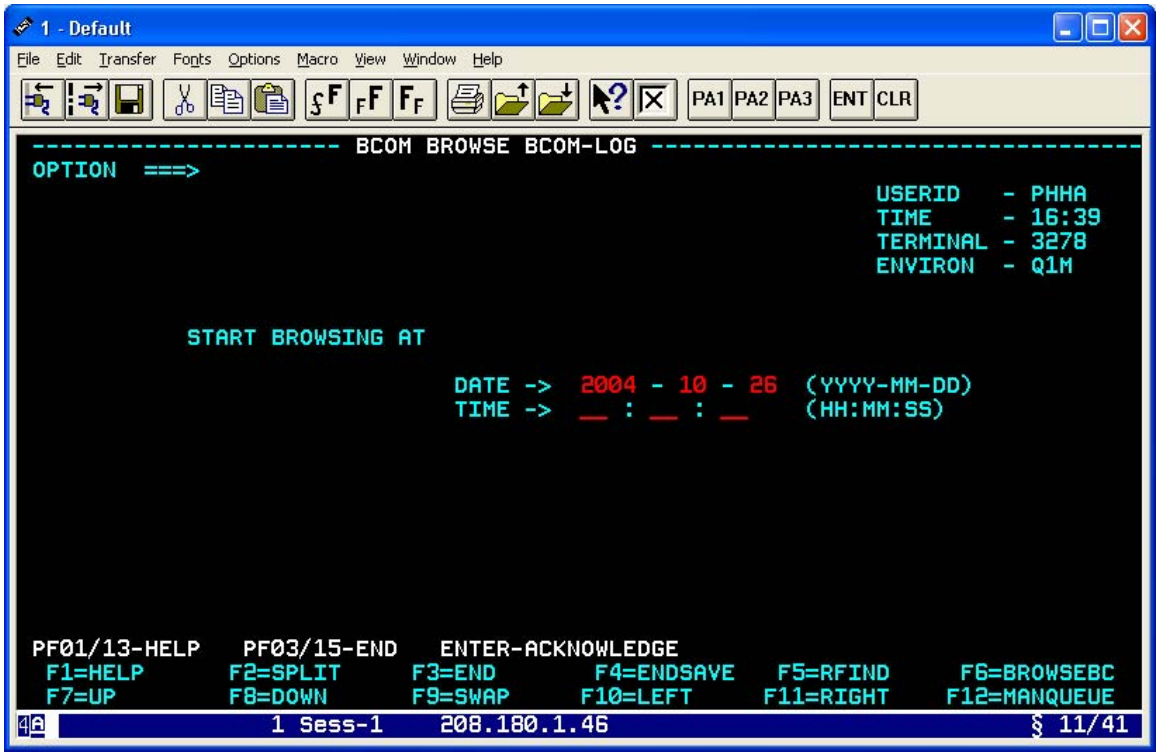
Please use MVS command CANCEL <BCOM-jobname> to stop the BCOM environment.

Browsing the BCOM Log

You can use this command to browse the log of BCOM messages generated by your local location.

For your convenience, the same BROWSE LOG command is also available from most other screens of Operational Commands.

Figure 4-18: Browse Log Command



Log Selection Criteria

If you select this command, you will be prompted for optional log selection criteria:

DATE If you want to specify a starting date for the BCOM log then enter as much of the date information as desired (YY, MM, DD). Note that the YEAR (YY) is required and that MM and DD will default to 01 if only the year is supplied.

HOUR If you want to specify a starting time for the BCOM log then enter as much of the time information as desired (HH, MM, SS).

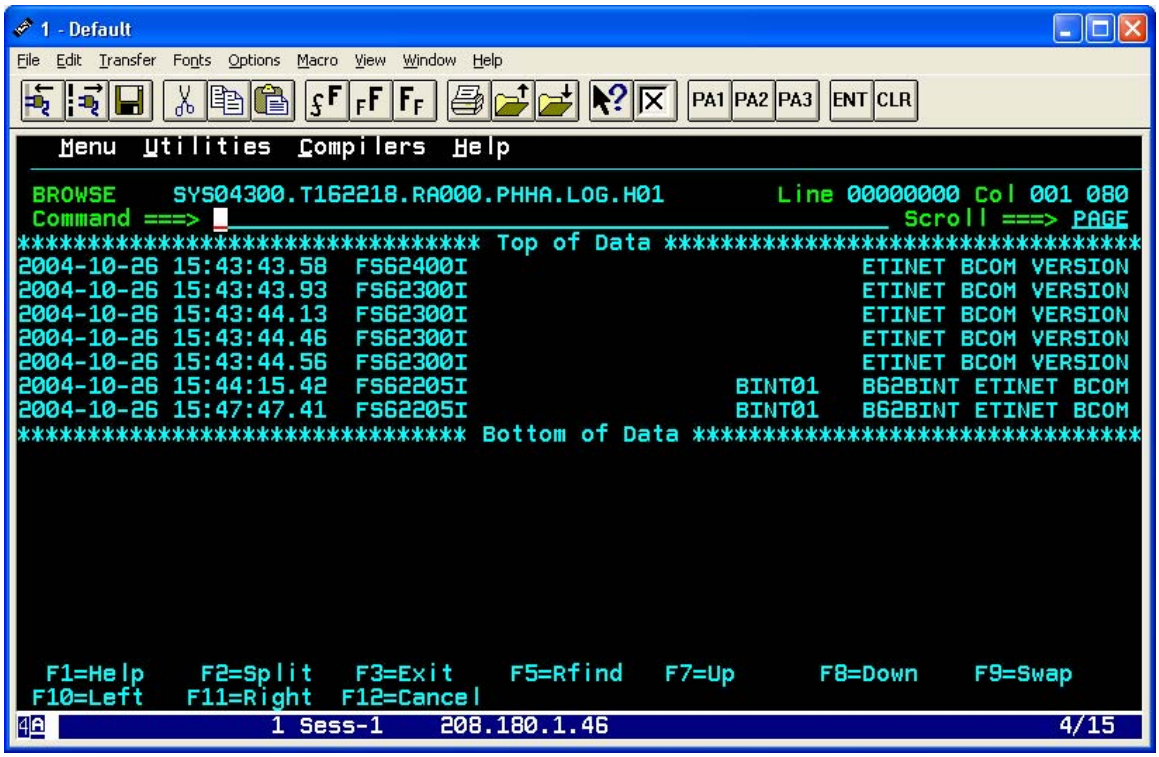
By default, if you omit the selection criteria, the User Interface will retrieve all records of the log starting from the current date.

When you have finished browsing the log (PF03=END) you will be returned to this same criteria selection screen and the DATE and HOUR fields will be set to your original selection or to today's date, if you had omitted the selection criteria.

Viewing the Results

The resulting log of messages is sent to a sequential dataset and can be browsed through standard ISPF commands. Examples of such commands include:

Figure 4-19: Log Command Results



P10/22 COLUMN Scroll to the left

You can type 'M' into the COMMAND field and press PF10/22 to position the display on the RIGHTMOST columns of the log.

P11/23 COLUMN Scroll to the right

You can type 'M' into the COMMAND field and press PF11/23 to position the display on the LEFTMOST columns of the log.

P08/20 FORWARD Scroll to next page

You can type 'M' into the COMMAND field and press PF08/20 to position the display on the LAST or BOTTOM page of the log.

P07/21 BACKWARD Scroll to previous page

You can type 'M' into the COMMAND field and press PF07/21 to position the display on the FIRST or TOP page of the log.

To search for a text item in the log, you can type the following TSO/ISPF command:

```
FIND  'xxxxxxxx'
```

into the COMMAND line at the top of the ISPF screen where 'xxxxxxxx' is your search argument. To locate the next occurrences of that same text string in the log (e.g. RFIND), simply press PF05/17 repeatedly.

Application Interface to CICS

A brief Overview

The BCOM Application Programmatic Interface (API) for CICS is a transactional interface for on-line CICS transactions to define and queue requests programmatically to BCOM.

The CICS API is invoked by the issuing an EXEC CICS LINK statement from within one of your transaction programs. The module of ETI-NET being called by this LINK statement will allocate a BINT conversation on the control session that has been established between CICS and BCOM at system initialization time.

The API's connection relies on APPC/LU6.2 protocols and parallel sessions to dramatically simplify any configuration requirements: it lets you concentrate on the BINT functions while the CICS and BCOM sub-systems automatically handle the allocation and management of multiple, concurrent conversations.

You can control the connection between CICS and BCOM from either side of the link through BCOM operator commands or through CICS operator commands, although the scope of control is not exactly the same.

The API structure

The BCOM API for CICS programs consists of one module called BCICSS01 and a common data area, which is exchanged between your Transaction Program and the ETI-NET module. The following paragraphs will describe the contents of this linkage section.

The Linkage Section

To use the Application Programming Interface (API) to CICS, your application program must allocate a linkage section in which it specifies the BCOM commands to submit and from which it can retrieve the resulting messages and completion statuses.

This linkage area consists of the following four sections:

- A FUNCTION AREA - To identify the type of call being made to the API routine;
- AN INPUT AREA - Contain the input data submitted to the CICS API;
- AN OUTPUT AREA - this section contains the result of the BCOM command;
- A CONVERSATION INFO AREA - this is an area used by the API to save context between calls.

The following paragraphs describe the contents of each section.

The Function Area

The BINT-FUNCTION is a 1-character field that describes the type of call being made to the CICS API. Your program can issue one of three different types of calls (or functions) to the CICS API of BCOM:

INITIALIZATION

This function allows the CICS API to establish a communicating session with BCOM; it is requested by entering the letter 'I' into the BINT-FUNCTION field.

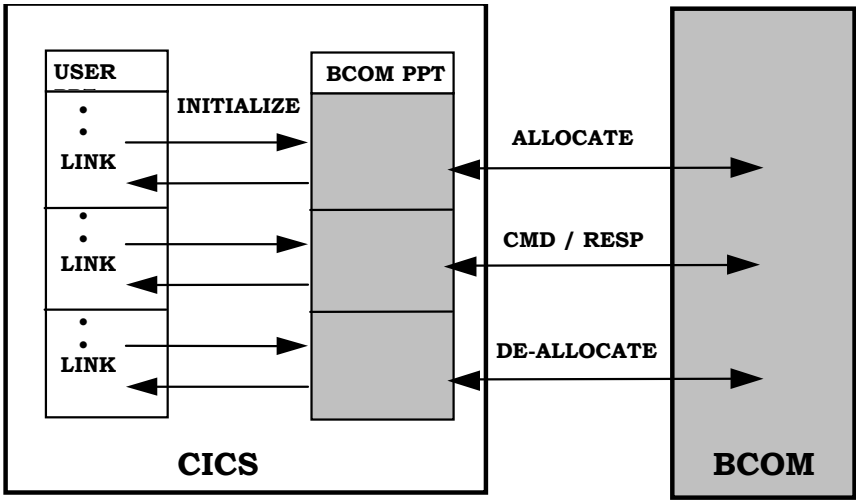
COMMAND

This function allows your program to submit a command to BCOM; it is requested by entering the letter 'C' into the BINT-FUNCTION field.

TERMINATE

this function is used to terminate your BINT conversation with BCOM; it is requested by entering the letter 'T' into the BINT-FUNCTION field.

Figure 5-1: API Overview



The Input-Area Section

The content of the Input area varies with the function requested:

INITIALIZATION
For function "I", your program uses the Input area section to specify the SYSID name of the CICS terminal definition (TCT) to be used in order to communicate with BCOM.

COMMAND
For function "C", your program uses the Input-Area section to enter a BINT command to BCOM (in command line format or as you would type into the line-mode BINT of TSO):

SET
To set keywords of a request definition.

RESET
To reset the keywords of a request definition.

ADD
To add a request to the request file.

QUEUE
To queue a request by name.

PURGE
To purge a request from the request file.

EXIT
To terminate processing.

Please consult *Appendix A - Command Syntax and Reference Summary* for more information on the particular syntax of these commands

TERMINATE
For function "T", this area is not used.

The Output-Area Section

The BINT command responses are stored in a buffer called AREA-B0. The size of this area can vary, depending on the type of command issued: your program must preset the AREA-LEN of BINT-OUTPUT-AREA to the maximum size of the buffer area, prior to calling the INITIALIZATION function. This value is then applied to all command responses returned during the BINT session.

BINT command responses are returned in a table format and in some cases the result may occur several times; to this effect, the NUM-LINES field is always set to the number of occurrences in the buffer.

The Conversation-Info Section

This section consists of a large context area into which the CICS-API routine saves the context of the current application's conversation with BCOM.

Your application should clear this field prior to invoking the "I" function.

Upon return from a successful "I" function call, your application must not modify the area until after the "T" function has completed.

Putting It All Together

Before we start discussing the details of the BCOM API to CICS, let us paint a global picture by describing a sample flow and its usage.

A Sample Flow

Typically, a user program would use the routine as follows:

- a first call would be made to INITIALIZE a connection to BCOM; this allocates a conversation with BCOM;
- a second call would be made to purge an existing request definition (e.g. created during a previous run);
- then the API would be called repeatedly to define a request and then add it to the request file of BCOM;
- the program would then call the API once more to queue the request;
- finally, a last call would be made to TERMINATE the BINT session - this allocates the current conversation with BCOM.

Using the API in CICS TPs

Here are two simple examples of how the BCOM API can be used by on-line transaction programs of CICS:

- An MVS on-line system updates a DB2 database using CICS transactions. A management transaction allows authorized personnel to schedule the batch transfer of that database to an SQL database on NSK Guardian.

Coding For The CICS-API

Any CICS transaction program can issue BCOM commands by calling the BCOM API for CICS.

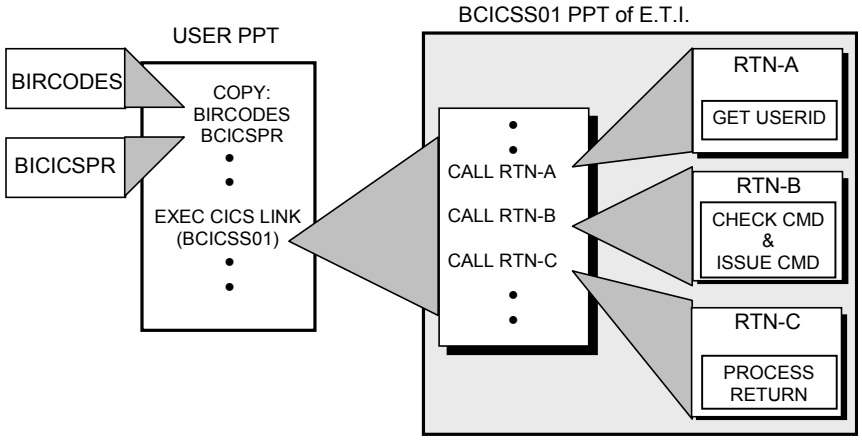
The API is implemented by an ETI-NET provided program called BCICSS01, which your application program invokes using an EXEC CICS LINK statement of CICS.

Program Construct

Supporting all calls to the BCICSS01 module are two copybooks which are supplied by ETI-NET in the BCOM installation tape and which your application program must source in:

- BICICSPR - this is a data structure that describes the parameters to the API routine and the response data returned;
- BIRCODES - this is the list of return codes and reason codes set by the API.

Figure 5-2: The API Structure



The BCICSS01 module uses the ISC facilities of CICS to send your commands to BCOM after translating them to an internal BINT format; the BCOM Monitor is responsible for processing the commands and returning a response.

When the response is received by the API, it is laid out into the output-area and the return codes are set according to the completion of the command; copybooks BICICSPR and BIRCODES define this completion information.

The Interface Parameters (BICICSPR)

The Interface parameter area consists of four distinct, as followed:

```
01 BCICS-PARAM.
   03 BINT-FUNCTION                PIC X(01).
      88 BINT-FUNCTION-INITIALIZE  VALUE 'I'.
      88 BINT-FUNCTION-COMMAND     VALUE 'C'.
      88 BINT-FUNCTION-TERMINATE   VALUE 'T'.

   03 BINT-COMMAND                 PIC X(80).
   03 INIT-COMMAND REDEFINES BINT-COMMAND.
      05 SYSID                     PIC X(04).
      05 FILLER                    PIC X(76).

   03 BINT-OUTPUT-AREA.
      05 RETURN-CODE-B0            PIC S9(04) COMP.
      05 REASON-CODE-B0            PIC S9(04) COMP.
      05 CONVID                    PIC X(04).
      05 NUM-LINES                 PIC S9(04) COMP.
      05 AREA-LEN                  PIC S9(04) COMP.
      05 AREA-B0                   PIC X(3000).
      05 WK-O-RTS-RECORD rEDEFINES AREA-B0.
         10 WK-O-RTS-DATA OCCURS 300 TIMES.
            15 WK-O-RTS-RET        PIC S9(04) COMP.
            15 WK-O-RTS-SUBJECT    PIC X(08).

   03 BCOM-CONVERSATION-INFO       PIC X(8198).
```

The following describes the fields of the CICS-API:

BINT FUNCTION

This field tells the BCICSS01 program whether to start or terminate a BINT session with BCOM or to send a command and process the response.

Functions supported are:

- **'I'** to INITIALIZE the API or start a BINT session
- **'C'** to send a command and get the response back
- **'T'** to terminate the BINT session with BCOM

BINT-COMMAND

When BINT-FUNCTION is valued “I”, your application supplies the name of the CICS SYSID (e.g. the SYSIDNT of the TCT for BCOM) to be used in communicating with the proper BCOM application, into the SYSID redefined area of the INIT-COMMAND field.

When BINT-FUNCTION is valued “C”, this input area contains a BCOM command formatted in the command-line mode syntax: this is as you would enter from a TSO terminal in line-mode (e.g. 'SET LOCAL-ATTR01 DISP=SHR’).

This field is not used when BINT-FUNCTION is valued “T” (FOR terminating the BINT session).

RETURN-CODE-B0

This field of the BINT-OUTPUT-AREA is set by the API to indicate whether the request was successful or not. If not zero, then your application should check the REASON-CODE-B0 field for details.

It should be initialized to 0 prior to calling BCICSS01.

REASON-CODE-B0

This field of the BINT-OUTPUT-AREA is set by the API to a non-zero value to explain the reason for a non-successful request or to indicate an action that has to be taken by your application program.

This field should be initialized to 0 prior to calling BCICSS01.

CONVID

This field of the BINT-OUTPUT-AREA identifies the particular SNA session to which your BINT conversation is allocated.

It is set by the API upon returning from a successful "I" function, must not be modified by your application program and must be supplied in all subsequent calls to the API - until a “T” function is performed and the conversation ID is deallocated.

NUM-LINES

This field of the BINT-OUTPUT-AREA indicates the total number of message lines received by the API and inserted into AREA-B0. See below for certain considerations.

AREA-LEN

When BINT-FUNCTION is valued “I”, this field serves to tell the BCOM API the maximum size of response data that your application is prepared to receive from BCOM following any BINT command.

When BINT-FUNCTION is not valued "I", this field of the BINT-OUTPUT-AREA is ignored.

AREA-B0

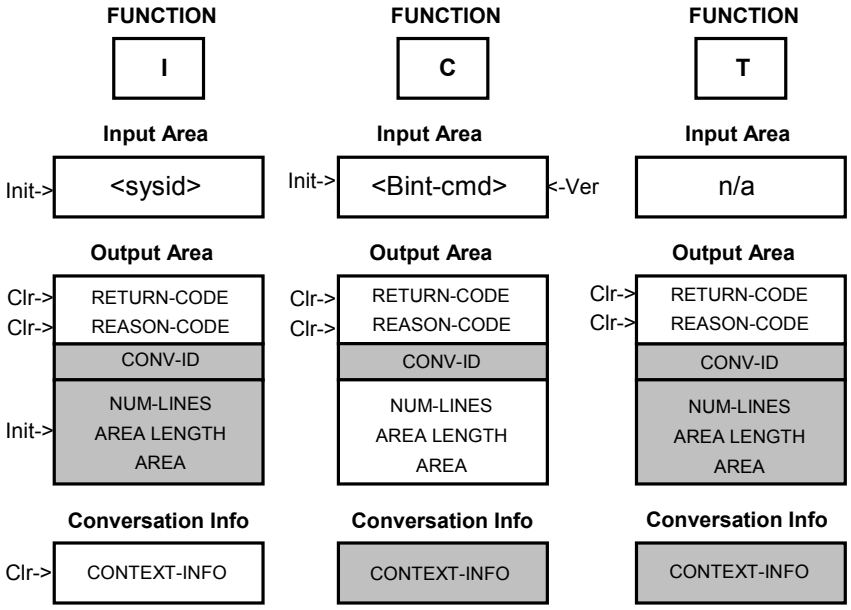
This field of the BINT-OUTPUT-AREA contains the BCOM response to your BINT command. See below for certain considerations.

BCOM-CONVERSATION-INFO

This area serves as context and work area to the BCICSS01 program while processing commands for your BINT session.

Your application must not modify this area until it has issued a "T" function to terminate the conversation with BCOM.

Figure 5-3 - The Linkage Section of the API



The 'Init' tags identify fields that must be initialized before calling the API; 'Clr' tags represent fields to be cleared or zeroed out prior to the call; 'Ver' tag represents a field to verify upon returning from the call.

Shaded areas represent either context areas that your program must not modify, or fields that are not related to the particular function being performed.

Invoking the API

After you have formatted the input/output linkage area, you invoke the API routine by issuing the following CICS command from your application program:

```
EXEC CICS LINK
   PROGRAM('BCICSS01')
   COMMAREA(<interface-data-structure>)
   LENGTH(<interface-data-structure-length>)
END-EXEC.
```

In this example, the field <interface-data-structure> refers to the BCICS-PARAM data structure.

This link expects the program 'BCICSS01' to be defined to CICS as a valid PPT entry. And as with any standard CICS command to pass control to a program, the following possible conditions may apply: NOTAUTH, PGMIDERR.

Transaction Security and the CICS-API

The BCOM product provides different levels of security signon, command and resource access security, as well as request ownership that are all based on the local and remote user identifiers (USERID).

The User Interfaces to BCOM (whether TSO Menu-based, the command-line interface, or this API for CICS programs) all comply with this principle by capturing the USERID under which their BINT connection has been established and sending that identifier to BCOM as part of initialization processing.

This also applies to your CICS transaction programs when they invoke the CICS API to BCOM-MVS in order to perform the BINT functions:

- as part of command-level security processing, this MVS userid will be checked for a master operator level in BCOM-MVS in order to determine the scope of commands allowed;

- this MVS USERID will be saved in your requests as part of request ownership, and verified for a match before allowing a PURGE command to execute;

- as for transfer processing, this same USERID will be used by BCOM-MVS to access local resources, and will be transmitted to the remote location where it will be correlated to a local ID under which access to local resources can take place.

Security processing is an option of the configuration file of BCOM. Suppose that the TP program that invokes the BCOM API for CICS is running under an uninitialized MVS userid (e.g. CSSN or CESN processing not done for the terminal that submits the program): in such a case, a blank USERID will be sent to BCOM and the request will fail at some point (depending on whether it is a command or a request) unless BCOM has been configured to run without security.

Processing The BCOM Response

Most commands of the BCOM User Interface return some form of message and completion codes. This information is stored in the RETURN-CODE, REASON-CODE and AREA-B0 fields of the API's output-area.

Assuming a positive return code from BCOM, let us see how a Transaction program will process the data found in AREA-B0.

The Size of AREA-B0

The response to a BINT command normally takes the form of one or more messages that describe the results of the command.

For programmatic access to BINT responses, these messages are converted into entries in a table and inserted by the API into the output buffer at AREA-B0. The maximum size of this output buffer area, as supplied by ETI-NET's BICICSPR copy book, is 3000 bytes: this size should be sufficient for all commands supported by the API1.

As a convention, your application must always tell the API just how much data it is prepared to accept in return for a command response (whether the 3,000 default value or some other number).

You do this by setting AREA-LEN to the maximum size of the AREA-B0 field prior to calling the API for processing the "I" (INITIALIZE) function.

Interpreting the AREA-B0 Data

On return from the call to BCICSS01, the structure of the data found inside AREA-B0 will depend on the type of command that was issued: the copy book BICICSPR includes RE-DEFINE statements for every type of command response that the API can return in the AREA-B0 field.

The size and contents of each table entry line is provided by these REDEFINES statements, and the number of entries in the table is given by NUM-LINES. NUM-LINES is adjusted to the size of each entry and the amount of storage available in AREA-B0: if AREA-B0 is too small, the API will insert as many entries as will fit into this area and will set the reason-code to REA-OUTAREA-TOO-SMALL.

Example of the Purge Command

The PURGE (and also the QUEUE) command syntax allows you to process one or more requests with a single command. The response message for a multi-request command generates a separate entry for each request name and its command completion status.

The copy book BICICSPR includes a definition called WK-0-RTS-RECORD that maps or re-defines the output-area for use with the response to a purge or queue command:

```
05 WK-0-RTS-RECORD REDEFINES AREA-B0.
   10 WK-O-RTS-DATA OCCURS 300 TIMES.
      15 WK-O-RTS-RET      PIC S9(04) COMP.
      15 WK-O-RTS-SUBJECT  PIC X(08).
```

There will be one WK-0-TRS-DATA entry per request purged; you will find the name of each request in sub-field WK-0-RTS-SUBJECT and the associated purge completion status in WK-0-RTS-RET.

Return & Reason Codes (BIRCODES)

The copybook BIRCODES also contains the list of possible RETURN-CODES and REASON-CODES set by the API for every command supported:

RETURN CODE	REASON-CODE	USAGE
RET-OK	RE-NO-REASON	all OK.
RET-WARNING	REA-SOME-REPEATING-ERRORS	the command and response apply to several objects, one of which, at least, is in error.
	REA-MUST-CALL-BINTTERM	you must now call TERMINATION.
	REA-EMPTY-FILE-INFO-CMD	INFO command made but Request file is empty.
	REA-QUEUE-FILE-EMPTY	QLIST command made but Queue file is empty.
	REA-OUTAREA-TOO-SMALL	Increase your AREA-B00 (PLUs the associated occurs clauses and fillers) beyond 3000 bytes.
	REA-NO-CMD-SUPPLIED	Command area supplied is empty.
	REA-MORE-THAN-1-OBJECT	List form of the command is not supported: only 1 object allowed.
RET-ERROR	see BIRCODES copy book	See BIRCODES copybook.

Interpreting the BCOM Codes

RETURN-CODE-B0 and REASON-CODE-B0 are global status fields that indicate the completion of a command issued to the API.

Some of the BINT commands - like EXIT, SET and RESET - only return these status fields while others - like the ADD, PURGE and QUEUE - also return a table of response messages. In the case of a multi-object command, the individual entries of the table will have to be inspected to determine the command objects, if any, that could not be processed.

- if the global RETURN-CODE-B0 contains zero, then all individual entries in the table list are OK;

- but if at least one of the objects in the table is in error (e.g. request name in a purge list is not found), the global RETURN-CODE-B0 will be set to RET-WARNING and the global REASON-CODE-B0 will be set to REA-SOME-REPEATING-ERROR.

Follow this procedure when inspecting the result table for individual entries. For each entry:

- if WK-0-RTS-RET contains a zero value then the command completion status is OK for that request;
- if WK-0-RTS-RET contains a non-zero value, then an error occurred while processing the command for that particular request. This non-zero value is itself mapped by the list of reason codes associated with the field REASON-CODE-B0.

Pseudo-Code Example

The following examples describe, in pseudo-code, how your user program can code for the three basic functions of the CICS-API.

Starting a BINT Session

When coding for the INITIALIZATION function:

(1) set the CICS system identifier in the input-area,
(2) clear the global status fields and conversation info area,
(3) set the area-len field to the maximum size of the expected response, and
(4) set the function code to “I” and link to BCICSS01:

```
MOVE 'I' TO BINT--FUNCTION
MOVE SYSID TO BCICS-PARAM
MOVE ZEROES TO RETURN-CODE-B0
MOVE ZEROES TO REASON-CODE-B0
MOVE some value to AREA-LEN
MOVE LOW-VALUES TO BCOM-CONVERSATION-INFO
EXEC CICS LINK PROGRAM('BCICSS01'),
     COMMAREA(BCICS-PARAM),
     LENGTH(BCICS-PARAM-LENGTH)
IF RETURN-CODE-B0 <> RET-OK
   PERFORM handle-error
ELSE
   continue. . .
```

If the CICS-BCOM connection has not been started, then a call to the 'Initialize' function will fail with reason code REA-INIT-CONN-ERR. An operator CEMT ACQUIRE connection will then be necessary.

Sending a BINT Command

When coding for the COMMAND function:

(1) move the command to the input-area,
(2) clear the global status fields
(3) set the function code to 'C' and link to BCICSS01:

```
MOVE 'C' TO BINT-FUNCTION
MOVE ZEROES TO RETURN-CODE-B0
MOVE ZEROES TO REASON-CODE-B0
MOVE command data to BINT-COMMAND

EXEC CICS LINK PROGRAM('BCICSS01'),
     COMMAREA(BCICS-PARAM),
     LENGTH(BCICS-PARAM-LENGTH)

IF RETURN-CODE-B0 = RET-OK then
   PERFORM command-OK
ELSEIF RETURN-CODE-B0 = RET-WARNING then
   IF REASON-CODE = REA-MUST-CALL-BINTTERM then
      PERFORM termination-sequence
   ELSE
      PERFORM handle-warning
ELSE PERFORM handle-error
```

Ending the BINT Session

When coding for the TERMINATE function:

(1) clear the global status fields
(2) set the function code to 'T' and link to BCICSS01:

```
MOVE 'T' TO BINT-FUNCTION
MOVE ZEROES TO RETURN-CODE-B0
MOVE ZEROES TO REASON-CODE-B0
EXEC CICS LINK PROGRAM('BCICSS01'),
     COMMAREA(BCICS-PARAM),
     LENGTH(BCICS-PARAM-LENGTH)

IF RETURN-CODE-B0 = RET-OK
   PERFORM end-program
 ELSE
   PERFORM report error
   PERFORM end-program
```

A Sample Program - APPLBINT

A complete COBOL program example is provided on the installation tape: member name APPLBINT of the source library of BCOM shows you how a CICS transaction program can be invoked to work as a BCOM User Interface to 3270 terminals owned by CICS.

The program prompts for some operational context, allocates a BINT conversation and then runs purely as a pass-through function to BCOM until such time as the terminal operator decides to terminate the BINT conversation.

Configuring For The API

VTAM Information

Configuring for the CICS-API requires the definition of a VTAM APPLID for CICS and for BCOM to be used for the BINT sessions.

This configuration is outside the scope of this manual; please refer to the *BCOM for MVS - Installation and Operations Guide* for more information about it.

Compile JCL

To compile your CICS program (or the sample program provided by ETI-NET), to use the CICS API to BCOM, you can use the sample JCL which is provided to that effect on the installation tape of BCOM; please refer to member APICJCL of the SOURCE distribution library of BCOM.

Operational Commands

BCOM operator commands can be used to control the BINT sessions established through the CICS API; CICS operator commands can also be used to control the BINT sessions established by CICS transaction programs. For more information, please refer to your *BCOM for MVS - Installation and Operations Guide*.

Application Interface for MVS

A brief Overview

The BCOM Application Programmatic Interface (API) for MVS is a callable interface for COBOL or ASSEMBLER programs to define and queue requests programmatically to BCOM.

The MVS API consists of a load module that must be link-edited to your program; using a defined interface convention, your program will cause this routine to allocate a BINT session with the Main task of BCOM-MVS and to process your BINT commands.

The API uses APPC/LU6.2 protocols to communicate with the Main Task of BCOM; it is designed to manage all aspects of session management with the BCOM sub-system such that your program can concentrate on the BINT functions at hand. A separate session is started for each MVS program that uses the API.

Meanwhile, operational commands are available to the BCOM operator in order to supervise or control the global connections of MVS programs to BCOM.

The API Structure

The BCOM API for MVS programs consists of a single load module called B62BINT, and a common data area: the latter is used to exchange commands and responses between your application and the BCOM Main Task.

The BINT Functions

The B62BINT module contains three callable functions that implement the interface to BCOM:

- BINTINIT - this function asks the API to establish a session with the Main Task of BCOM;
- BINTCMD - this function allows your program to submit a command to BCOM over that BINT session;
- BINTTERM - this function is used to terminate the BINT session with BCOM.

These three subroutines execute non-reentrant, non-reusable code because they store and maintain context for the BINT session in progress.

The Linkage Section

Your application program must provide a linkage section for the MVS API with which to exchange BINT commands and BINT responses with BCOM.

Additional information - such as the command completion status - will be returned by the API inside appropriate fields of this linkage section.

There are three areas to the linkage section:

- INITPARM - this area serves to establish an initial session with the Main Task, using certain configurable defaults which will be discussed later on;
- INPARM - this area contains the input commands to be submitted to BCOM;
- OUTPARM - this area contains the result of the BCOM command and its completion status.

The following table maps the areas of the linkage section onto the three functions of the BCOM programmatic interface. They are identified as Pn parameters where n represents the calling order:

FUNCTION	PURPOSE	INITPARM	INPARM	OUTPARM
BINTINIT	START BINT SESSION	P1	n/a	P2
BINTCMD	SUBMIT COMMAND	n/a	P1	P2
BINTTERM	END BINT SESSION	n/a	n/a	P1

INITPARM Area Details

The INITPARM area is used to identify: the type of API function being used, the parameters that specify the connection between the API and BCOM, and the default request definition keywords that are in effect during this BINT session.

The MVS API Function

MVS Assembler and COBOL programs will use the 'User Application Programmatic Interface' (UAPI) of B62BINT. Hence, the string 'UAPI' must be entered into the INIT-NAME field of the INITPARM area.

The API-BCOM Connection

The connection between the API and the Main Task is established and maintained using SNA protocols. This requires that you supply the following VTAM Application Identifiers (APPLID):

- INIT-MY-ID - enter into this field the APPLID which your MVS program will use when calling the API;
- INIT-FS62-ID - enter into this field the APPLID used by the Main Task of BCOM.

Session Default Parameters

Your program can simplify the task of defining BCOM transfer requests by pre-setting certain keywords to some initial value. The API then substitutes those default values when the corresponding fields of a request definition are not supplied by your program.

The default parameters supported are:

LOCAL-LOC
: this is the default name for the LOCAL-LOCATION keyword in a request definition; name starts by the symbol %;

REMOTE-LOC
: this is the default name for the REMOTE-LOCATION keyword of a request definition; name starts by the symbol %;

PRIORITY-CLASS
: this is the default value for the PRIORITY-CLASS keyword of a request definition;

ROUTING-CLASS
: this is the default value for the ROUTING-CLASS keyword of a request definition.

INPARM Area Details

The IN-COMMAND area of INPARM is the field in which your program submits a BINT command for BCOM processing.

The BCOM API for MVS programs expects BINT commands to be entered in a standard BCOM command format; for example, the following SET command defines the name of a file at a remote NSK Guardian location:

```
SET  REMOTE-FILE  $DATA01.SUBVOL.FILE
```

The IN-COMMAND field has a maximum size of 80 bytes; if the command issued is shorter than 80 bytes, then the remaining field positions must be padded with blanks.

OUTPARM Area Details

Generally speaking, the OUTPARM area contains the response data to a BINT command or API function, together with control fields that identify the completion status for that function call.

Other fields of the OUTPARM area must be set prior to calling the BINT function, in order that both the API and your application program can equally parse the BINT command response.

The Completion Status Fields

The global result of the function invoked is indicated by the field OUT-RETURN.

If OUT-RETURN indicates an error, then the field OUT-REASON will provide its details. More details on these fields, later.

The Output-Area Section

The BINT command responses are stored into a buffer called OUT-AREA. The size of this area can vary depending on the type of command issued: your program must preset the OUT-AREA-LEN of OUTPARM to the maximum size of the buffer area, prior to calling the BINTINIT function. This value is then applied to all command responses returned during this BINT session.

BINT command responses are returned in a table format, because, in some cases, the result may occur several times. The API always sets the OUT-LINES field to the number of occurrences in the buffer. This is described, in more detail, in a later paragraph.

Putting it all together

The following table summarizes the actions to be taken by your program against the above fields, prior to calling the MVS API.

Fields	BINTINIT	BINTCMD	BINTTERM
INIT-NAME	set 'UAPI'		
INIT-MY-ID	program's APPLID		
INIT-FS62-ID	BCOM's APPLID		
INIT-LOCAL-LOC	(optional %location)		
INIT-REMOTE-LOC	(optional %location)		
INIT-ROUTING-CLASS	(optional class value)		
INIT-PRIORITY-CLASS	(optional class value)		
IN-COMMAND	n/a	BINT command	
OUT-RETURN	zeroes		
OUT-REASON	zeroes		
OUT-LINES	zeroes	zeroes	
OUT-AREA-LEN	3000[1]		
OUT-AREA	spaces	spaces	spaces

[1] Specify the size of the PIC X area that your program is using in the BINTCOPY book. We recommend 3000.

Userid Security and the MVS-API

The BCOM product provides the following levels of security: signon, command and resource access security, as well as request ownership. These are all based on the local and the remote user identifiers (or USERIDs).

All User Interfaces to BCOM comply with this principle by capturing the USERID under which their BINT connection has been established and sending that identifier to BCOM as part of initialization processing:

- as part of command-level security processing, the MVS userid associated to your program will be checked for a master operator level in BCOM-MVS in order to determine the scope of commands allowed;
- the MVS userid associated to your program will be saved in your requests as part of request ownership, and verified for a match before allowing a PURGE command to execute;
- as for transfer processing, the MVS userid associated to your program will be used by BCOM-MVS to access local resources, and will be transmitted to the remote location where it will be correlated to a local ID under which access to local resources can take place.

Security processing is an option of the configuration file of BCOM.

Processing the BCOM Response

All commands of the BCOM User Interface return some form of message and completion codes. This information is stored into the OUT-RETURN, OUT-REASON and OUT-AREA fields of the OUTPARM area.

The Size of OUT-AREA

To simplify the programmatic access to BINT responses, response messages are converted into table entries that are inserted by the API into table structures that map the output buffer at OUT-AREA. The maximum size of this output buffer area, as supplied by ETI-NET's BINTCOPY copy book, is currently 3000 bytes: this size should be sufficient for all commands supported by the API3.

As explained earlier, to inform the API of how much data your application is prepared to accept in return for a command response (whether the 3,000 default value or some other number), you must set the OUT-AREA-LEN field to the appropriate value before calling the BININIT function.

Interpreting the Response

On return from a call to the API, the table inserted into the OUT-AREA field is set according to the type of command that was issued. The copybook BINTCOPY includes the necessary REDEFINE clauses for every type of command response supported.

The number of entries in the response table is given by OUT-LINES: this parameter is adjusted to the size of each entry and the amount of storage available in OUT-AREA according to the initial value of OUT-AREA-LEN: if OUT-AREA is too small, the API will insert as many entries as will fit into this area and will then set the OUT-REASON to REA-OUTAREA-TOO-SMALL.

Example of the Purge Command

The PURGE (and also the QUEUE) command syntax allows you to process one or more requests with a single command. The response message for a multi-request command generates a separate entry for each request name and its command completion status (e.g. request purged, request not-found, security-violation, etc).

The copy book BINTCOPY includes a definition called WK-0-RTS-RECORD that maps or re-defines the output-area for use with the response to a purge or queue command:

```
05 WK-0-RTS-RECORD REDEFINES OUT-AREA.
   10 WK-0-RTS-DATA OCCURS 300 TIMES.
      15 WK-0-RTS-RET      PIC S9(4) COMP.
      15 WK-0-RTS-SUBJECT  PIC X(8).
```

WK-0-RTS-RECORD maps the field OUT-AREA; you will find a single WK-0-RTS-DATA entry for each request referenced by the purge command; within this entry, sub-field WK-0-RTS-SUBJECT will identify the name of the request and sub-field WK-0-RTS-RET will indicate its purge completion status.

Return and Reason Codes (BIRCODES)

The copybook BIRCODES contains the list of the possible OUT-RETURN codes and OUT-REASON codes that can be set by the API for each command supported. Some of the more frequent codes are:

OUT-RETURN	OUT-REASON	USAGE
RET-OK	RE-NO-REASON	All OK.
RET-WARNING	REA-SOME-REPEATING-ERRORS	The command and response apply to several objects, one of which, at least, is in error.
	REA-MUST-CALL-BINTTERM	You must now call TERMINATION.
	REA-EMPTY-FILE-INFO-CMD	INFO command made but Request file is empty.
	REA-QUEUE-FILE-EMPTY	QLIST command made but Queue file is empty.
	REA-OUTAREA-TOO-SMALL	Increase your AREA-B00 (PLUs the associated occurs clauses and fillers) beyond 3000 bytes.
	REA-NO-CMD-SUPPLIED	Command area supplied is empty.
	REA-MORE-THAN-1-OBJECT	List form of the command is not supported: only 1 object allowed.
RET-ERROR	See BIRCODES copy book	See BIRCODES copy book.

Interpreting the Completion Codes

The OUT-RETURN and OUT-REASON fields are global status indicators for the completion of a command issued to the API.

Some of the BINT commands - like EXIT, SET and RESET - only return the global status indicators; other commands - like the ADD, PURGE and QUEUE - also return a table of response messages. If multi items are returned in the table, a global code will indicate to your program that it must inspect each entry to determine the command object in error.

In Summary

- if the global OUT-RETURN contains zero, then all individual entries in the table list are OK;

- if at least one of the objects in the table is in error (e.g. request name in a purge list is not found), the global OUT-RETURN will be set to RET-WARNING and the global OUT-REASON will be set to REA-SOME-REPEATING-ERROR.

If you must inspect the result table for individual entries, then follow this procedure:

- if WK-0-RTS-RET contains a zero value then the command completion status is OK for that request;

- if WK-0-RTS-RET contains a non-zero value, then an error occurred while processing the command for that particular request. This non-zero value will be mapped by the list of reason codes associated with the field OUT-REASON.

Pseudo-Code Example

The following examples describe, in pseudo-code, how your application program can use the API for an MVS interface to BCOM.

Starting a BINT Session

When coding for the INITIALIZATION function:

(1) set the API type identifier to 'UAPI',
(2) clear the global status fields and output area,
(3) set the area-len field to the maximum size of the expected response
(4) optionally supply some default request definition parameters, and
(5) set the appropriate APPLID names:

```
MOVE 'UAPI' TO INIT-NAME
MOVE program-APPLID TO INIT-MY-ID
MOVE bcom-APPLID TO INIT-FS62-ID
[ MOVE default-value TO INIT-LOCAL-LOC ]
[ MOVE default-value TO INIT-REMOTE-LOC ]
[ MOVE default-value TO INIT-ROUTING-CLASS ]
[ MOVE default-value TO INIT-PRORITY-CLASS ]
MOVE ZEROS TO OUT-RETURN
MOVE ZEROS TO OUT-REASON
MOVE some value to OUT-AREA-LEN
MOVE SPACES TO OUT-AREA
CALL 'BINTINIT'  USING  INITPARM
                        OUTPARM
IF OUT-RETURN <> RET-OK THEN
   PERFORM handle-error
ELSE
   continue...
```

If the connection to BCOM cannot be established, then a call to the 'Initialize' function will fail with reason code REA-INIT-CONN-ERR.

Sending a BINT Command

When coding for the COMMAND function:

(1) move the command to the IN-COMMAND field,
(2) clear the global status fields as well as the OUT-LINES field and the output area:

```
MOVE SPACES TO OUT-AREA
MOVE ZEROS TO OUT-RETURN
MOVE ZEROS TO OUT-REASON
MOVE ZEROS TO OUT-LINES
MOVE command data to IN-COMMAND

CALL 'BINTCMD' USING INPARM
                     OUTPARM

IF OUT-RETURN = RET-OK then
   PERFORM command-OK
ELSEIF OUT-RETURN = RET-WARNING then
   IF OUT-REASON = REA-MUST-CALL-BINTTERM then
      PERFORM termination-sequence
   ELSE
      PERFORM handle-warning
ELSE
   PERFORM handle-error
```

Ending the BINT Session

When coding for the TERMINATE function: clear the global status fields and the output area:

```
MOVE SPACES TO OUT-AREA
MOVE ZEROS TO OUT-RETURN
MOVE ZEROS TO OUT-REASON

CALL 'BINTTERM'  USING OUTPARM

IF OUT-RETURN = RET-OK then
   PERFORM end-program
ELSE
   PERFORM report error
   PERFORM end-program
```

A Sample Program - B62APISP

A complete COBOL program example is provided on the installation tape: member name B62APISP of the source library of BCOM shows you how a COBOL MVS program can get a dataset name from the JCL parm and list all the requests in the request file of BCOM that have this particular dataset as a local-file name.

Configuring for the API

VTAM Information

Configuring for the MVS-API requires only the definition of the VTAM APPLID that will be used by your MVS program in order to establish a BINT session with BCOM.

The APPLID of programs that use the MVS API should be defined the same as those used for regular TSO BINT sessions. Please refer to the *BCOM for MVS - Installation and Operations Guide* for additional information.

Link-Editing JCL

Compile your COBOL program (or the sample program provided by ETI-NET) and link-edit the MVS API to BCOM using the sample JCL provided to that effect on the installation tape of BCOM; please refer to member APIMJCL of the SOURCE distribution library of BCOM.

```
//STEP01 EXEC COBVSCL
//  PARM . COB='LOAD, BUF(50K),SIZE=256K,CLIST,
//                  NORESIDENT,NODYNAM,VERB'
//COB .  SYSPRINT DD SYSOUT=*
//COB .  SYSLIB   DD DSN=YOUR.BM.INSTALIB,DISP=SHR
//COB .  SYSIN    DD DSN=YOURMVS.INSTALIB(B62APISP), DISP=SHR
//LKED . SYSLMOD  DD DSN=YOURMVS.APPLICAT.LIB, DISP=SHR
//LKED . SYSLMOE  DD DSN=YOUR.MVS.LOADLIB, DISP=SHR
//LKED . SYSIN    DD *
   NAME  B62APISP
   INCLUDE  SYSLMOD(B62APISP)
   INCLUDE  SYSLMOD(B62BINT)
   ENTRY  B62APISP
   ORDER  B62APISP
   NAME  B62API(R)
/*
//
```

Operational Commands

BCOM operator commands can be used to monitor the BINT server sessions established through the MVS API. For more information, please refer to the *BCOM for MVS – Installation and Operations Guide*.

Defining BCOM Requests

This section is organized according to the following sub-sections:

1. **Request/Queue Architecture**

   Give an overview of Requests and Queues, and describes the relationship between them.

2. **Request Definition Commands**

   Provide an overview of the BINT 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**

   Outline the operations that can be performed to manage requests from within a BINT session. It discusses the commands required to do this and includes examples of how they are used.

4. **Queue Control Facilities**

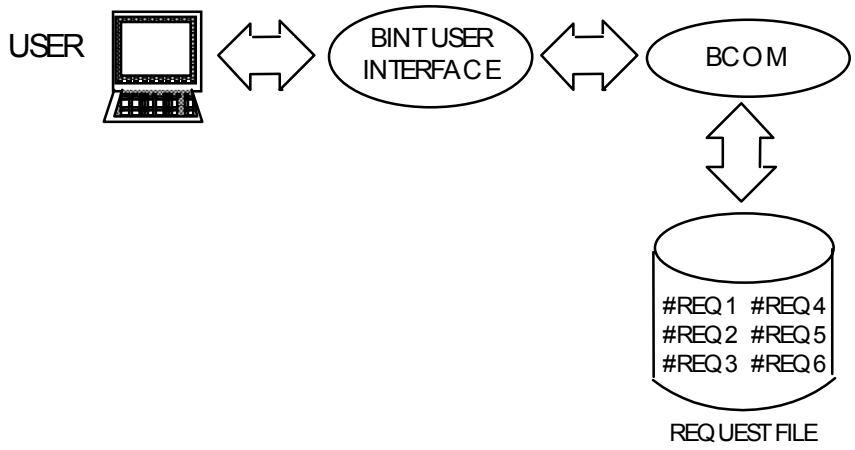
   Discuss the operations that can be performed from within a BINT session to manage queued requests. It discusses the individual commands needed to do this, and provides examples of how they can be used.

The Request / Queue Architecture

Request Basics

Transfer requests are defined by BCOM users through the BINT user interface. The definitions of these transfer requests are stored on the BCOM Request file. After all transfer attributes have been described to the BINT user interface using a series of SET commands, the request may be added to the Request file. This is accomplished via the BINT ADD command.

Figure 7-1

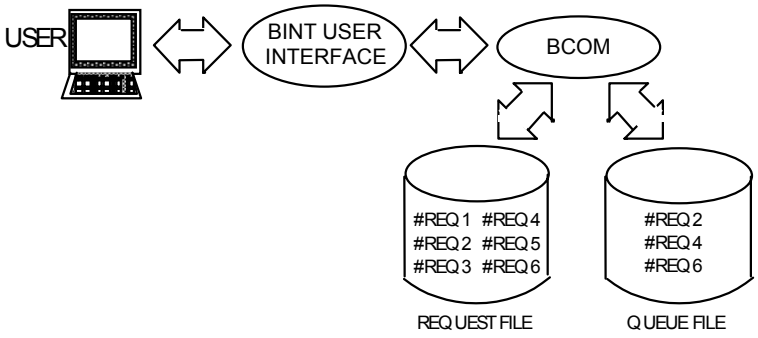


Requests are maintained on the request file until explicitly purged by the user.

Queue Basics

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

Figure 7-2



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. (One exception is when using the Checkpoint Facility. When checkpointing and the transfer is not successful, the queue file entry is not deleted, but held for Restart. Another exception is scheduling with a repetition factor. Another exception is when auto restart is specified, and a line failure during the transfer occurs).*

The outcome of the transfer can be ascertained by means of a BINT STATUS request-name command. See the sub-section 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 sub-section provides an introduction to the different aspects of Request Definition processing using the BINT 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 A - Command Syntax and Reference Summary*.

BINT Request Definition Commands

The BINT 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.

NOTE: For correct individual command syntax, please refer to *Appendix A - Command Syntax and Reference Summary* in this manual.

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: from-location, to-location, request type, from-file information, to-file information, request options, etc.... It is used with options, as follows:

SET LOCAL-LOCATION logical-location-name

Indicate the location of the current BINT process. Qualify the local-file name that must be defined in the Configuration File and begins with a '%' sign.

SET REMOTE-LOCATION logical-location-name

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

SET TYPE request-type

Indicate type of the request. As followed

SET TYPE RQ

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

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 Obey, etc.). If the job executes locally, then the type is a LJ Local request supported on MVS otherwise the type is RJ (Remote-job).

SET TYPE RB | RR

Indicate that this is request for executing a backup (RB) or restore (RR) request whose definition is in a remote BCOM platform Request File.

SET TYPE SP | RP (for transfers between MVS's only)

Indicate that this is a PDS to PDS file transfer type request. SP (Send PDS) is an outbound transfer request and RF (Receive PDS) is an inbound request.

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 platforms, 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 further discussions on ROUTING-CLASS support, please refer to the *BCOM for MVS - Installation and Operations Guide*.

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 re-transmitted 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.

> ***Note:*** *The combination SET TYPE LJ with SET LOCAL-FILE is not supported at this time.*

SET REMOTE-FILE file-name

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

SET LOCAL-ATTRnn

Define the attributes of the local-file, as required by the type of transfer request defined.

Examples of LOCAL-FILE-ATTRnn values on MVS platforms are: DCB, DISP, LABEL, UNIT, 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.

SET REMOTE-ATTRnn

Define the attributes of the remote file, as required for the type of request defined. The nn is a consecutive number which starts at 01 and increases for each attribute to be defined.

Examples of REMOTE-ATTRnn values 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.

> ***Note:*** *The syntax and supported values of the LOCAL-ATTRnn and REMOTE-ATTRnn sub-parameters are fully depicted in the* Appendix A*, - SET sub-section.*

Reviewing the Definitions

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 BINT Session State.

The session states guarantee that current values for all SET commands will be "remembered" by the BINT interface. Thus, you can SET and RESET BINT 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.

The individual BINT Session State Commands are described below.

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

The RESET Command

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

When the BINT 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 pre-defined information items and attributes. With SHOW, the user can verify all the BINT 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 BINT session. When a BINT 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

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 BINT Session State.

Now that requests have been added to the Request file, on the basis of one logical record definition per request, 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 in individual sub-sections below. Actual examples of how requests might be managed and controlled are then presented.

> ***Note:*** *For correct individual command syntax, please refer to* Appendix A - 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 sub-set 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

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 BINT Session State.

Display File Transfer Attributes

To display file transfer attributes for the #EXAMP21 request:

```
INFO  #EXAMP21
REQUEST = #RQSTNAME
TYPE = RECEIVE-FILE
RQST - COMMENTS:
01  THIS IS THE COMMENT LINE
ROUTING CLASS = 19
PRIORITY-CLASS = 05
LOCAL-LOCATION = %MVSMTL
LOCAL-FILE = TEST.DOWNLOAD.DATA
LOCAL-FILE-ATTRIBUTES:
01  DISP=SHR
REMOTE-LOCATION = %TAND1
REMOTE-FILE = $DATA01.SVOL.TANFILE
REMOTE-FILE-ATTRIBUTES:
01  PROCESS-PRIORITY 150
AUTO-RESTART = NO
CHECKPOINT   = 1500
COMPRESSION  = NO
ABNORMAL-END = NONE
NORMAL-END   = NONE
OWNER ID     = OWNER-ID
```

Display Request Names

To display all request names:

```
LISTREQ
 #EXAMP10  #EXAMP11  #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
BMSG003 - CMD PURGE FOR THE REQUEST #EXAMP10 COMPLETED SUCCESSFULLY
```

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
TYPE = RECEIVE-FILE
RQST - 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 = 07
PRIORITY-CLASS = 05
LOCAL-LOCATION = %MVSMTL
LOCAL-FILE = TEST.DOWNLOAD.DATA
LOCAL-FILE-ATTRIBUTES:
01  DISP=SHR
REMOTE-LOCATION = %TAND1
REMOTE-FILE = $DATA01.SVOL.TANFILE
REMOTE-FILE-ATTRIBUTES:
01  PROCESS-PRIORITY  150
AUTO-RESTART = NO
CHECKPOINT   = 1500
COMPRESSION  = NO
ABNORMAL-END = NONE
NORMAL-END   = NONE
OWNER ID     = OWNER-ID
```

In this example, we can see the modified routing class (07) on request definition 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);
- Restart a previously abended request running with Checkpoint (QUEUERESTART);
- Queue interactively (QUEUEWAIT);
- Display the status of queued requests (STATUS).

QUEUE Command

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

The ROUTING-CLASS and PRIORITY-CLASS can also be specified when a request is queued.

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. Note that ABORT can also be on a transfer process.

It is not recommended that you stop a file transfer request while it is executing, since the destination file may be left in a corrupted state. However, it is possible to do so by using the ABORT command with the name of the request to be stopped. This form of the ABORT command is only available with BCOM.

The ABORT command can also be used to ABORT specific transfer processes configured under a particular ROUTING CLASS. Please refer to the Operations section of the *BCOM for MVS - Installation and Operations Guide* for details on the ABORT facility.

The 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 the Request file). A request that has been held does not lose its rank in the queue. It can be released at any time by using the QUEUEREL command.

The 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 BINT process to queue a request and wait for its completion before returning to process your next command. This is useful when you wish to include the queuing of a request within a BCOM Obey file (NSK Guardian-specific), where the remaining commands must not be executed prior to completion of the transfer.

The QUEUERESTART Command

When a request is queued with CHECKPOINT and it fails (perhaps due to a lost LU or Transfer Process abend), the Queue file entry will remain on the Queue file in a "HOLD for RESTART" state. Use the QUEUERESTART command to restart the transfer from the last commit point. This is useful for large transfers since there is no re-transmission of "committed" records.

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.

> ***Note:*** *If an old version of the Request file is restored from a backup, this information may not reflect the latest execution of the non-queued requests.*

Queue Management Examples

Aborting an Executing File Transfer

```
ABORT #EXAMP1
BMSG003 - CMD ABORT FOR THE REQUEST #EXAMP1 COMPLETED
SUCCESSFULLY
```

Deleting a Request from the Queue

```
QUEUEDEL  #EXAMP1
BMSG003 - CMD QUEUEDEL FOR THE REQUEST #EXAMP1 COMPLETED SUCCESSFULLY
```

Holding and Releasing a Request

```
QUEUEHOLD  #EXAMP1
BMSG - CMD QUEUEHOLD FOR THE REQUEST #EXAMP1 COMPLETED SUCCESSFULLY

QLIST
```

REQUEST	PRI	QUEUE TIME	STATUS	RLOC	RTECLAS	CHKPNT	RSTRT
#EXAMP1	05	1998-03-31 11:55:19	HELD	%TDMMTL	02	0000	No
#EXAMP2	04	1998-03-31 11:55:20	EXECUTING	%TDMMTL	01	0000	No
#EXAMP3	02	1998-03-31 11:59:49	WAITING	%TDMTOR	02	0000	No

```
QUEUEREL  #EXAMP1
BMSG - CMD QUEUEREL FOR THE REQUEST #EXAMP1 COMPLETED SUCCESSFULLY

QLIST
```

REQUEST	PRI	QUEUE TIME	STATUS	RLOC	RTECLAS	CHKPNT	RSTRT
#EXAMP1	05	1998-03-31 11:55:19	EXECUTING	%TDMMTL	02	0000	No
#EXAMP3	02	1998-03-31 11:59:49	WAITING	%TDMTOR	02	0000	No

Display the List of Queued Requests

```
QLIST
```

REQUEST	PRI	QUEUE TIME	STATUS	RLOC	RTECLAS	CHKPNT	RSTRT
#EXAMP1	05	1998-03-31 11:55:19	HELD	%TDMMTL	02	0000	No
#EXAMP2	04	1998-03-31 11:55:20	EXECUTING	%TDMMTL	01	0000	No
#EXAMP3	02	1998-03-31 11:59:49	WAITING	%TDMTOR	02	0000	No

Interactive Queuing

```
QUEUEWAIT  #EXAMP1
BMSG - CMD QUEUEWAIT FOR THE REQUEST #EXAMP1 COMPLETED
SUCCESSFULLY
```

Display the Execution Status of a Request

```
STATUS  #EXAMP1
REQUEST=#EXAMP1
QUEUE  TIME = 1998-03-31  11:55:19
START  TIME = 1998-03-31  12:01:23
END  TIME = 1998-03-31  12:03:34
RECORD  COUNT = 200
BLOCK  COUNT = 5
MESSAGE = TRANSFER  SUCCESSFUL
```

The message area is the primary means of establishing the outcome of the completed transfer. The following is a list of possible values that may appear in the MESSAGE area:

```
TRANSFER SUCCESSFUL
EXECUTING
SELECTED NOW
WAITING FOR EXECUTION
DELETED BY OPERATOR
HELD BY OPERATOR
ERROR  HELD FOR RESTART
ERROR  HELD FOR AUTO RESTART
SYSTEM ERROR
NOT EXECUTED YET
WAITING FOR SHEDULING
SUCCESSFUL **WAITING
SUCCESSFUL **W-SHED
REMOTE ALLOCATION ERROR
LOACAL ALLOCATION ERROR
REMOTE SUB-TASK ABEND
REMOTE OPEN ERROR
LOCAL OPEN ERROR
TRANSFER ABENDED
REMOTE PROCESSING ERROR
USER NOT CORRELATED
DUPLICATE REMOTE REQUEST
REMOTE REQUEST NOT FOUND
LOCAL SUB-TASK ABEND
ABORTED BY OPERATOR
REMOTE SEC VIOLATION
REMOTE USER NOT FOUND
REMOTE CHECKSUM ERROR
REMOTE SEC IMPL ERROR
LOCAL SEC VIOLATION
LOCAL USER NOT FOUND
LOCAL CHECKSUM ERROR
LOCAL SEC IMPL ERROR
LOST LU
REMOTE QUEUE NOT SUPPORTED
```

BCOM File Transfers

This section discusses 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 section is organized according to the following sub-sections:

1. Procedures for Defining File Transfer Requests explain how to send files from MVS platforms and how to receive files from remote locations.

2. Data Conversion Utilities explain how to perform selectable field data translation.

3. Check-Point / Re-Start Facilities explain how to associate and use checkpointing information on a file transfer request.

Procedures for Defining File Transfer Requests

The examples included on the following pages give step-by-step outlines of the procedures involved in defining file transfer requests.

The examples cover a reasonable cross-section of the kind of cases most shops are likely to encounter.

For in-depth explanations of the individual syntax rules for the BINT commands, please refer to *Appendix A - Command Syntax and Reference Summary*.

Request Definition Overview

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

Step I: Define the Request

1. **Get ready**

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

2. **Set the request attributes**

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

3. **Add the request**

   Within the same BINT 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

1. **Identify the queue overrides**

   This step is optional.

2. **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 processed in a First In First Out (FIFO) sequence within the PRIORITY-CLASS unless otherwise indicated.

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

Transfers to Remote Platforms

File Transfers originating on the MVS may be targeted to BCOM-supported remote platforms. Information specific to a particular target platform will be captioned as such, indicating that the documentation applies to transfers from the MVS origin to that particular remote platform.

The BCOM architecture strives for as much platform independence as possible. Particular differences a platform may exhibit will be illustrated in the captioned sections. Examples will reinforce the transfer type as well as the particular details pertaining to the source and destination platforms.

The procedural descriptions below use the structure outlined above (Step I, Step II) to describe how to define transfer requests.

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. Two examples that illustrate these steps are then provided.

The first example assumes that the BINT process through which the file transfer is defined, currently executes at an MVS location; 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.

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 *BCOM for NSK Guardian - User's Guide* for specific information.

1. Get ready

In order to define a request that involves a transfer from MVS to NSK Guardian, the following information must be known:

MVS Information:

The local file: the local 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:

	Required MVS FROM-ATTRs for Transfers from an Existing MVS Dataset	
	DISK	**TAPE**
Data Control Block (DCB)	Optional	Optional
Disposition (DISP)	Mandatory	Mandatory
Dataset Name (DSN)	Mandatory	Optional*
Label	--------	Optional*
Space	--------	--------
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 (abend code 3060) if any record has a LRECL greater than 4096.

NSK Guardian Information:

The remote file: The remote 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. pre-allocated).
- 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.

Load options

When the remote file is an ENSCRIBE disk file, FUP is used as the I/O process. The options described in the Tandem FUP manual for the LOAD command are applicable for a BCOM transfer whose REMOTE-LOCATION is Tandem. See the SET REMOTE-ATTRnn PARMS attribute in the example.

I/O process priority

BCOM assigns a default priority to the I/O module that will be responsible for writing to the local file. You may alter this priority for a specific request.

General Information:

Field conversion

BCOM converts the data from EBCDIC to ASCII when transferring records from MVS to NSK Guardian. It is important to know the position and length of the binary fields where conversion override is required. Refer to the *Data Conversion Utilities* in this chapter for more details.

2. Set the request attributes

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

General Information:

```
SET  TYPE             SF
SET  REMOTE-LOCATION  logical-location-name
SET  REMOTE-FILE      NSK Guardian File Name
SET  REMOTE-ATTRnn    attributes
SET  LOCAL-LOCATION   logical-location-name
SET  LOCAL-FILE       MVS File Name
SET  LOCAL-ATTRnn     attributes
SET  PRIORITY-CLASS   priority-value
SET  ROUTING-CLASS    routing-class-value
SET  COMMENTnn        text-comments
SET  NORMAL-END       #request-name
SET  ABNORMAL-END     #request-name
SET  CHECKPOINT       checkpoint-count
```

Notes:

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

2. *For further information on the SET CHECKPOINT command and its usage refer to the CheckPoint/Restart Facilities section.*

NSK Guardian Information:

```
SET  REMOTE-FILE      NSK Guardian-file-name
SET  REMOTE-ATTR01    PROCESS-PRIORITY=priority-value
```

MVS Information:

```
SET  LOCAL-FILE       MVS-dataset-name
SET  LOCAL-ATTRnn     DCB=MVS DCB info
SET  LOCAL-ATTRnn     DISP=dataset disposition
SET  LOCAL-ATTRnn     LABEL=MVS tape label info
SET  LOCAL-ATTRnn     SPACE=MVS space info
SET  LOCAL-ATTRnn     UNIT=MVS unit info
SET  LOCAL-ATTRnn     VOL=MVS volume info
```

3. Add the request

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

```
ADD  #request-name
```

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 BINT session or at a later time. In this case, the local file must be present on MVS. Remote disc files must be present when the destination is NSK Guardian. If it is a remote process, it must be running at the time the request is queued.

1. Identify the queue overrides

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

a. Alter the PRIORITY-CLASS of a request which has already been defined on the Request file and change its PRIORITY-CLASS value.

b. Alter the ROUTING-CLASS of a request that has already been defined on the Request file and change its ROUTING-CLASS value.

2. 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.

MVS to NSK Guardian; Example 1

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

Step I: Define the Request

1. Get ready

- The local dataset on MVS already exists, its name is EXAMP1.DSN, and the disposition is share.
- The remote file on NSK Guardian is a key-sequenced file named $DATA01.MYSUB.DATA.
- The data is to be loaded into this file with 25% slack space in each block as specified in the PARAMS designator of the REMOTE-ATTR command.
- We let BCOM assign the priority to the NSK Guardian I/O process.
- The ROUTING-CLASS used for the transfer will be 01.

2. Set the request attributes

```
READY
CALLBINT
SET  TYPE              SF
SET  LOCAL-LOCATION    %MVSMTL
SET  REMOTE-LOCATION   %TAND1
SET  ROUTING-CLASS     01
SET  LOCAL-FILE        EXAMP1.DSN
SET  LOCAL-ATTR01      DISP=SHR
SET  REMOTE-FILE       $DATA01.MYSUB.DATA
SET  REMOTE-ATTR01     PARAMS = DSLACK  25
```

3. Add the request

```
ADD  #REQDN01
```

Step II: Activate the Request

1. Identify the queue overrides

There are no specific overrides in this example.

2. Queue the Request

```
QUEUE  #REQDN01
```

MVS to NSK Guardian; Example 2

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

Step I: Define the Request

1. Get ready

- The local file has the DCB of a print file.
- The disposition of this file is old.
- The PRIORITY-CLASS that the queued request will have is 2.
- The local dataset name is "EXAMPLE.NO1."
- The file is stored as the first dataset sequence number on the labelled tape.
- The UNIT name is TAPE.
- The remote file is a process named $PROG.
- The BCOM I/O process priority must be 170.
- The ROUTING-CLASS required is 03.
- Default EBCDIC to ASCII conversion.

2. Set the request attributes

```
READY
CALLBINT
SET  TYPE              SF
SET  LOCAL-LOCATION    %MVSMTL
SET  REMOTE-LOCATION   %TAND1
SET  ROUTING-CLASS     03
SET  LOCAL-FILE        EXAMPLE.NO1
SET  REMOTE-FILE       $PROG
SET  PRIORITY-CLASS    2
SET  LOCAL-ATTR01      DISP=OLD
SET  LOCAL-ATTR02      DSN=EXAMPLE.NO1
SET  LOCAL-ATTR03      LABEL=(1, SL)
SET  REMOTE-ATTR01     PROCESS-PRIORITY = 170
```

3. Add the request

```
ADD  #REQDN02
```

Step II: Activate the Request

1. Identify the queue overrides

None for this example.

2. Queue the request

```
QUEUE  #REQDN02
```

Defining NSK Guardian to MVS File Transfer Requests

Steps I and II below provide an outline of how to define an NSK Guardian to MVS file transfer request initiated from the MVS platform (e.g. a ReceiveFile with the MVS platform as the LOCAL location and the NSK Guardian platform as the REMOTE location). Two examples applying these steps are then provided.

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 re-use 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.

1. Get ready

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

NSK Guardian Information:

The Remote file: The remote 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.
- If a named process is to act as the source file, its process name must be known. Refer to the section on "Program-matic Interfaces" in the *BCOM for NSK Guardian User's Guide* for further details on NSK Guardian process support. Source file or process does not need to exist at this point.

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

The logical identifier for the NSK Guardian 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.

MVS Information:

The Local file

The local 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:

	Required MVS TO-ATTR's for Transfers from an Existing MVS Dataset	
	DISK	**TAPE**
Data Control Block (DCB)	Optional	Optional
Disposition (DISP)	Mandatory	Mandatory
Dataset Name (DSN)	Mandatory	Optional*
Label	--------	Optional*
Space	--------	Optional
Unit	Optional	Optional
Volume (VOL)	Optional	Optional

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

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.

General Information:

Field conversion

BCOM converts the data from EBCDIC to ASCII when transferring records from MVS to NSK Guardian. It is important to know the position and the length of any binary fields that must not be converted.

Request context

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).

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, BCOM will perform the transfer, but any record greater than 4096 will be truncated to 4096. If the NSK Guardian record size is shorter than the MVS record size, records will be padded with blanks up to the 256th character. Beyond the 256th character, the pad characters are unpredictable.

2. Set the request attributes

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

General Information:

```
SET  TYPE             RF
SET  LOCAL-LOCATION   logical-location-name
SET  LOCAL-FILE       MVS file name
SET  LOCAL-ATTRnn     MVS file attributes
SET  REMOTE-LOCATION  logical-location-name
SET  REMOTE-FILE      Guardian file name
SET  REMOTE-ATTRnn    remote file attributes
SET  PRIORITY-CLASS   priority-value
SET  ROUTING-CLASS    routing-class-value
SET  COMMENTnn        text-comments
SET  NORMAL-END       #request-name
SET  ABNORMAL-END     #request-name
SET  CHECKPOINT       checkpoint-count
```

Notes:

1. *For further information on the SET (AB)NORMAL-END command and its usage, refer to Follow-on Scheduling Chapter.*

2. *For further information on the SET CHECKPOINT command and its usage refer to the CheckPoint/Restart Facilities Section.*

NSK Guardian Information

```
SET  REMOTE-FILE      Guardian-file-name
SET  REMOTE-ATTR1     PROCESS-PRIORITY = priority-value
```

MVS Information

```
SET  LOCAL-FILE       MVS-dataset-name
SET  LOCAL-ATTRnn     DCB = MVS DCB info
SET  LOCAL-ATTRnn     DISP = dataset disposition
SET  LOCAL-ATTRnn     VOL = (MVS volume(s))
SET  LOCAL-ATTRnn     LABEL = MVS tape label info
SET  LOCAL-ATTRnn     SPACE = MVS space info
SET  LOCAL-ATTRnn     UNIT = MVS unit info
```

3. Add the request

Immediately following the SET commands (within the same BINT 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 BINT session or at a later time. The source file must be present on NSK Guardian. If an NSK Guardian process is supplying the data for the transfer, it must be running at the time the request is queued.

1. Identify the queue overrides

This step is optional. For example, the user may wish to:

a. RESTART a request that has been run with the CHECKPOINT option and has abended.

b. Alter the PRIORITY-CLASS of a request that has already been defined on the Request file and change its PRIORITY-CLASS value.

c. Alter the ROUTING-CLASS of a request that has already been defined on the Request file and change its ROUTING-CLASS value.

2. 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 in the same priority class 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.*

Receive NSK Guardian on MVS; Example 1

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

Step I: Define the Request

1. Get ready

- The remote file name is $DATA01.MYSUB.DATA.
- We let BCOM assign the priority to the NSK Guardian I/O process.
- 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 local file on MVS already exists. Its dataset name is EXAMP1.DSN, and the disposition is Share.
- The priority of this request is 3.
- The remote and local locations are an NSK Guardian labeled TAND1 and an MVS labeled MVSMTL. Both these names must be configured as location names in the NSK Guardian Configuration File. By default, the local-location name is assumed by the BINT process, but can be overridden.
- The priority of the BCOM READ I/O process will be 150.

2. Set the request attributes

```
READY
CALLBINT
SET  TYPE              RF
SET  LOCAL-LOCATION    %MVSMTL
SET  LOCAL-FILE        EXAMP1.DSN
SET  LOCAL-ATTR01      DISP=SHR
SET  REMOTE-LOCATION   %TAND1
SET  REMOTE-FILE       $DATA01.MYSUB.DATA
SET  REMOTE-ATTR01     PROCESS-PRIORITY = 150
SET  ROUTING-CLASS     06
SET  CHECKPOINT        50
SET  PRIORITY-CLASS    3
```

3. Add the request

```
ADD  #REQUP01
```

Step II: Activate the Request

1. Define the queue overrides

There are no specific overrides in this example.

2. Queue the request

```
QUEUE  #REQUP01
```

NSK Guardian to MVS; Example 2

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

Step I: Define the Request

1. Get ready

- The remote process name will be $PROG.
- The BCOM I/O process priority will be 170.
- The local file on MVS is a tape file. This file does not exist at the time the transfer is activated.
- The file has the DCB of a print file.
- The disposition of this file is NEW. If the transfer is successful, this file will be catalogued. However if the transfer is not successful, the file will be deleted.
- The local tape dataset name is "EXAMPLE.NO1."
- The file will be stored as the first file sequence on the tape.
- The tape UNIT name is "TAPE."
- ASCII to EBCDIC conversion will be performed on all fields (default).
- The remote and local locations are an NSK Guardian labeled TAND1 and an MVS labeled MVSMTL. Both these names must be configured as location names in the NSK Guardian and MVS Configuration File. In the MVS Configuration file, TAND1 appears as one of the REMOTE locations.

2. Set the request attributes

```
READY
CALLBINT
SET  TYPE             RF
SET  LOCAL-LOCATION   %MVSMTL
SET  LOCAL-FILE       EXAMPLE.NO1
SET  LOCAL-ATTR01     DCB=(LRECL=133,BLKSIZE=13300,RECFM=FBA)
SET  LOCAL-ATTR02     DISP=(NEW,CATLG,DELETE)
SET  LOCAL-ATTR03     DSN=EXAMPLE.NO1
SET  LOCAL-ATTR04     LABEL=(1,SL)
SET  LOCAL-ATTR05     UNIT=TAPE
SET  REMOTE-LOCATION  %TAND1
SET  REMOTE-FILE      $PROG
SET  REMOTE-ATTR01   PROCESS-PRIORITY = 170
```

3. Add the request

```
ADD   #REQUP02
```

Step II: Activate the Request

1. Define the queue overrides

None for this example.

2. Queue the request

```
QUEUE   #REQUP02
```

Notes:

1. *To process variable-length datasets, the data length of each record must be known. A 4-byte length field is then added to every logical record - or LRECL. These four additional bytes must be accounted for when you define the MVS DCB: for example, if the largest record size is n, then calculate the LRECL to be n+4.*

To help process a block of data that contains variable-length records, a similar 4-byte control field is added. In addition, the block must be large enough to accept the largest record size - or LRECL - and its own 4 bytes of control information: in other words, the size of the largest record + 8. These additional bytes must be accounted for in the MVS DCB.

Suppose that the largest logical record size you need to transfer is 4072 bytes, then defining the DCB with (RECFM+VB, LRECL=4076, BLKSIZE=4080) will allow the data to be stored with integrity.

2. *After the ATTRnn designator, syntax is identical to MVS JCL conventions.*

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

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, Tandem implements the ASCII character set, while IBM/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 the NSK Guardian, 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 (zero) is valid.

Length indicates the number of characters from the offset position that 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
```

The RESET field removes all field specifications.

Conversion Suppression - Example

The following examples illustrate the conversion suppression facility using the SET FIELD command.

```
SET  TYPE             SF
SET  REMOTE-LOCATION  %TAND1
SET  LOCAL-LOCATION   %MVSMTL
SET  REMOTE-FILE      $DATA01.SUBV.TEST1
SET  LOCAL-FILE       MVS.SOURCE.FILE
SET  LOCAL-ATTR01     DISP = SHR
SET  ROUTING-CLASS    3
SET  COMMENT01        THIS IS A COMMENT
SET  COMMENT02        THIS IS ANOTHER COMMENT
SET  FIELD            8,2,12,4
ADD  #RQSTNO1
```

Explanation:

In the above example, a request is being made to send a file from MVS to NSK Guardian. 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 re-transmitted in the event of transfer abend. This facility is very useful, particularly when large files must be transferred over slower speed lines.

How it Works

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 re-started from the beginning. The SET CHECKPOINT option of the request definition can be used to prevent this unnecessary re-transmission. When specified, it will guarantee that only that portion of the file after the last checkpointed value will be re-transmitted.

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 QUEUERESTART 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 QUEUERESTART 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 Tandem Unstructured files.

- THE QUEUERESTART 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.

- To restart the transfer from square one - e.g. from the beginning of the file to transfer - the keyword COLD can be added to the QUEUERESTART 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 recommencing the transfer.

- If the QUEUERESTART 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 in 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 QUEUERESTART, incoming records are compared against these records "written since last checkpoint" and dropped if equal; otherwise BCOM assumes it can recommence 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 an NSK Guardian location, the BCOM I/O process will start writing records at the last successfully checkpointed record. For requests originating from an NSK Guardian location, the BCOM I/O process will skip any records passed to it by the user's process until the checkpoint value is reached. Only then will BCOM recommence transmission starting from the last successfully checkpointed record.

- On MVS, users should be conscious of the DISP setting when the REMOTE-LOCATION is specified as MVS. If the target MVS file already exists, it is recommended that users specify SET REMOTE-ATTRnn DISP = SHR. If the target MVS file does not exist, users should specify SET REMOTE-ATTRnn DISP = (NEW, CATLG, CATLG). This is to ensure that the contents of the uncompleted transfer are preserved after abnormal termination of the original checkpointed request.

- Checkpoint-Restart does not apply to PDS transfers.

Considerations for Partitioned Datasets (PDS)

Two attributes are specific to the 'SP' (Send PDS) and 'RP' (Receive PDS) types of transfer. These types are defined for transfers from one partitioned data set (PDS) to another, between MVS's.

Members of partitioned datasets are selected or excluded using the special attributes "SELECT" and "EXCLUDE". A third attribute called "COPYPARM" controls how members are to be copied.

`SELECT | EXCLUDE`

The SELECT or EXCLUDE attribute can only be used as a local attribute with an 'SP' request, and as a remote attribute with an 'RP' request. It allows the user to designate which member(s) of the sending platform's library will be transferred.

The SELECT attribute designates specifically which member(s) will be transferred, whereas the EXCLUDE attribute designates which member(s) of a library will not be sent. For example:

`SELECT=(A,B,C,D)` A, B, C and D members are to be transferred.

or

`EXCLUDE=(Y,Z)` all members of the library aer to be sent, except for Y and Z.

Note 1: *You can repeat the SELECT or EXCLUDE statement as a separate ATTRnn attribute if the list of members is too long.*

Note 2: *SELECT and EXCLUDE are mutually exclusive. Therefore, you can use one or the other within a request, but not both.*

COPYPARM

The COPYPARM attribute can only be used as a local attribute with an 'SP' request, and as a remote attribute with an 'RP' request. This attribute allows the user, first to indicate whether the designated member(s) to be transferred will replace or not the member(s) in the receiving library, and also whether a list of the transferred members will be available in the BCOM log, or not.

Note that, when giving a NOREPLACE value to the COPYPARM attribute, you can, for a same library, designate some members to be replaced and some other not, by adding 'R' parameters in the SELECT attribute. For example:

```
SET .... COPYPARM=(NOREPLACE)
SET .... SELECT=(A,B,(C,,R),D,(X,,R))
```

...will trigger the transfer of the members A, B, and D without replacement and C and X with a replace indicator.

Note 1: *The default values for the COPYPARM attribute are Replace and Nolist; the syntax of this attribute is:*

```
COPYPARM=(REPLACE |NOREPLACE,LIST |NOLIST )
```

Note 2: *Using a DISP=OLD option with a Send-PDS request will lock the entire PDS, not only members*

Example

```
SET  TYPE             SP
SET  LOCAL-LOCATION   %MVS1
SET  REMOTE-LOCATION  %MVS2
SET  ROUTING-CLASS    01
SET  LOCAL-FILE       P34701A.PDS.LIB
SET  LOCAL-ATTR01     DISP=SHR
SET  LOCAL-ATTR02     SELECT=(D,E,F)
SET  LOCAL-ATTR03     COPYPARM=(NOREPLACE,LIST)
SET  REMOTE-FILE      P34701A.PDS.TARGET
SET  REMOTE-ATTR01    DISP=(SCR,CATLG,DELETE)
SET  REMOTE-ATTR02    UNIT=SYSDA
SET  REMOTE-ATTR03    DCB=(DSORG=PO,LRECL=80,BLKSIZE=4000,RECFM=80)
SET  REMOTE-ATTR04    SPACE=(TRK,(5,5,3))
```

Job Submission

This section 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 sub-sections:

1. Overview of Job Submission describes the BCOM usage conventions and guidelines for submitting jobs from the local location to a particular remote location.

2. Procedures for Local Job Submission explain how to import MVS JCL to execute jobs on the local MVS platform and how to submit JCL's residing on the local platform.

3. Procedures for Remote Job Submission explain how to export jobs to be run on a remote platform, and how to submit jobs residing on remote platforms.

Overview

This section describes the BCOM usage conventions and guidelines for submitting jobs from the local location to a particular remote location. More detailed implementation considerations for each type will be treated in separate sub-sections.

A JOB submission 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 JOB submission request you want to execute is already defined, you just have to queue the request.

Jobs can be submitted from any platform to execute either locally or on a remote platform. The job varies across platforms. On MVS platform, the JOB is a collection of JCL organized into a job stream. On Tandem platform, the JOB is a collection of commands suitable for processing by a TACL program. This is usually called an Obey file.

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

For a disk file, the fully qualified disk file name is required.

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

Where the JOB runs

The JOB may run locally or on a remote node. The TYPE=LJ (Local Job) and TYPE=RJ (Remote-Job) keywords are used to identify where the job will be executed. The LOCAL-LOCATION and REMOTE-LOCATION keywords are used to identify the name of the location at the platform on which it will be executed.

Remote Job Submission

The JOB submission request is initiated from the local platform; the JOB file data resides either locally or resides on the target platform; the JOB executes at the remote location.

Local Job Submission

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

About the JOB file

The JOB file can reside locally or reside on a remote node. The LOCAL-FILE and REMOTE-FILE keywords are used to identify the JOB input data; and the LOCAL-LOCATION and REMOTE-LOCATION keywords are used to identify the name of the node or location on which this file resides.

The chart below illustrates the combinations of parameters possible for the different types of Job Submissions profiles, as defined from the MVS platform:

PARAMETER	LOCAL JOB DATA=local	LOCAL JOB DATA=remote	REMOTE-job DATA=local	REMOTE-job DATA=remote
TYPE	LJ	LJ	RJ	RJ
local-location	R	R	R	R
local-file	R	-	R	-
local-attr	O	-	O	-
remote-location	R (must be the same as Local-Location)	R	R	R
remote-file	-	R	-	R
remote-attr	-	O	O	O
routing-class	R (must be '00')	R	R	R
priority-class	R	R	R	R
compression	must be NO	O	O	O
checkpoint	must be ZERO	O	O	O

Where:

R = parameter required
O = optional

Figure 9-1: Remote Job Submission

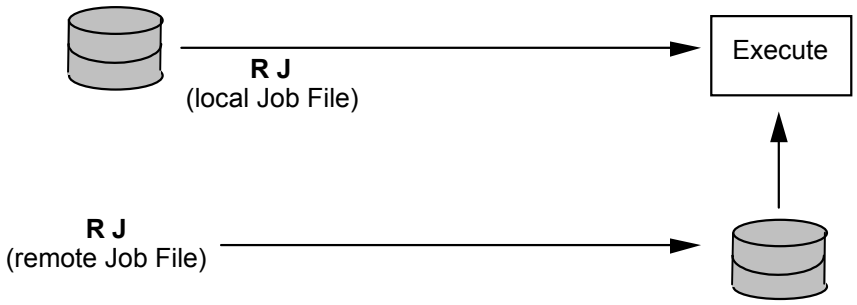
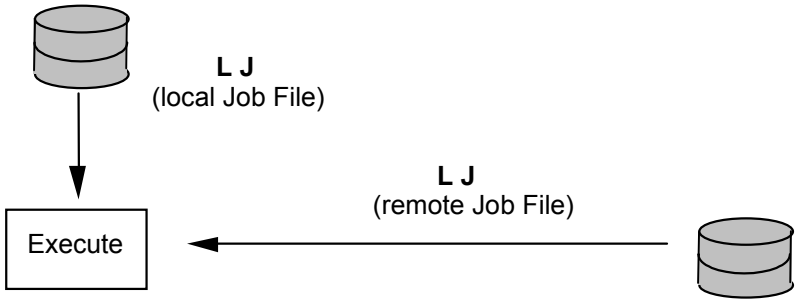


Figure 9-2: Local Job Submission



Procedures for Local Job Submission

This section discusses the BCOM JOB submission facility as it pertains to jobs executing on the MVS platform.

MVS JCL imported from a remote platform may be submitted on the local MVS platform or an MVS JCL of the local MVS platform may be submitted locally.

A request which is defined as a LOCAL-JOB will execute the JCL locally.

JCL Submission to MVS

Step I: Define the Request

An MVS JCL submission 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 JCL submission request you want to execute is already defined, you do not have to go through Step I.

1. Get ready

In order to define a JCL submission request, the following information must be known:

About the JCL: The JCL can reside either on the remote site or the local site. Use the REMOTE-FILE or LOCAL-FILE descriptor, as required.

About the Internal Reader: The JCL will be submitted to the internal reader. It is dynamically allocated for each request.

MVS Information:

LOCAL-LOCATION is a keyword used to define where the JCL will be executed.

General Information:

Request context

BCOM allows you to specify the next request name to be executed upon the completion of this Obey file submission. The next request can be of any type (SF, RF, LJ, RJ) and a different one can be chosen depending on the outcome of the ancestor request. The chained request may be selected based on the completion status of the request (normal or abnormal).

Note that the completion status of a JCL submission request only relates to the submission itself; the normal or abnormal end of any step included in this JCL is not handled by BCOM.

2. Set the Request Attributes

Using the information from Step 1, you can now enter the request definition using the BCOM Interface (BINT) program.

a. The following SET commands are used to define an MVS local job that uses a remote file of JCL:

General Information:

```
SET  TYPE             LJ
SET  PRIORITY-CLASS   priority-class-value
```

MVS Information:

```
SET  LOCAL-LOCATION   %MVS-location-name
```

Remote Information:

```
SET  REMOTE-LOCATION  %logical-remote-location-name
SET  ROUTING-CLASS    routing-class-value
SET  REMOTE-ATTR01    remote-attributes
SET  REMOTE-FILE      remote-file-name
```

b. The following SET commands are used to define an MVS local job that submits local JCL data:

General Information:

```
SET  TYPE             LJ
SET  PRIORITY-CLASS   priority-class-value
```

MVS Information:

```
SET  LOCAL-LOCATION   %MVS-local-location-name
SET  LOCAL-FILE       <name of the JCL File >
SET  LOCAL-ATTR01     Local-File Attributes
```

Since Local Job with Local File does not need to transfer any file and the submission process is done locally, only limited REMOTE information is needed with specific values:

Remote Information:

```
SET  REMOTE-LOCATION  must be the same as Local-Location
SET  ROUTING-CLASS    must be '00'
SET  COMPRESSION      must be  NO
```

3. Add the request

Immediately following the SET commands (within the same BINT 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 BINT session or at a later time. The JCL must be present on the appropriate node.

1. **Identify the queue overrides**

2. **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 in the same priority class will be processed first.

Figure 9-3: Local Job Submission with Remote File

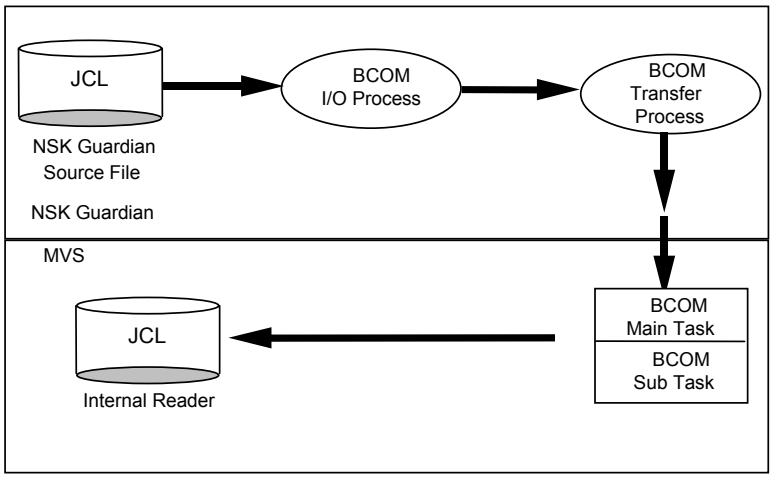
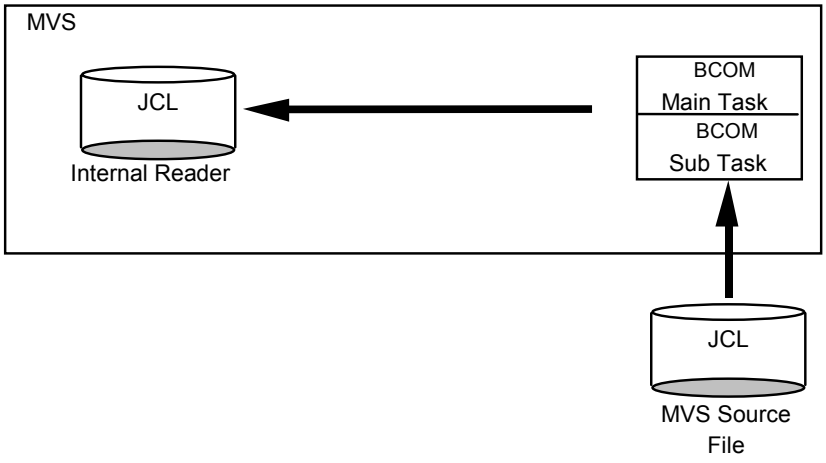


Figure 9-4: Local Job Submission with Local File



Local Job Submission - Example 1

Submit to the local MVS, JCL that resides on an NSK Guardian disk file.

Step I: Define the Request

1. Get ready

- The JCL is contained in a disc file named $DATA01.MYJOB.JOB.
- The ROUTING-CLASS over which the transfer will be initiated is set to 03.

2. Set the request attributes

```
READY
CALLBINT
SET  TYPE              LJ
SET  ROUTING-CLASS     03
SET  REMOTE-LOCATION   %TAND1
SET  LOCAL-LOCATION    %MVSMTL
SET  REMOTE-FILE       $DATA01.MYJOB.JOB
SET  PRIORITY-CLASS    5
```

3. Add the request

```
ADD #REQJCL1
```

Step II: Activate the Request

1. Identify the queue overrides

There is no need to define any queue overrides.

2. Queue the request

```
QUEUE  #REQJCL1
```

Local Job Submission - Example 2

Submit on MVS, a JOB generated by an NSK Guardian named process

Step I: Define the Request

1. Get ready

- The JCL is generated by a process named $GNJOB
- Set the ROUTING-CLASS to 02.
- We let BCOM assign the priority to the NSK Guardian I/O process.

2. Set the request attributes

```
READY
CALLBINT
SET  TYPE              LJ
SET  ROUTING-CLASS     03
SET  LOCAL-LOCATION    %TAND1
SET  REMOTE-LOCATION   %MVSMTL
SET  LOCAL-FILE        $GNJOB
SET  PRIORITY-CLASS    5
```

3. Add the request

```
ADD  #REQOB2
```

Step II: Activate the Request

1. Identify the queue overrides

There is no need to define any queue overrides.

2. Queue the request:

```
QUEUE  #REQOB2
```

Local Job Submission - Example 3

Submit to the local MVS, JCL that resides on a local disk file.

Step I: Define the Request

1. Get ready

The JCL is contained in a dataset named MVS.JCL.FILE.

2. Set the request attributes

```
READY
CALLBINT
SET  TYPE             LJ
SET  ROUTING-CLASS    00
SET  COMPRESSION      NO
RESET CHECKPOINT
SET  LOCAL-LOCATION   %MVSMTL
SET  LOCAL-FILE       MVS.JCL.FILE
SET  LOCAL-ATTR01     DISP=SHR
SET  REMOTE-LOCATION  %MVSMTL
SET  PRIORITY-CLASS   5
```

3. Add the request

```
ADD  #REQLJLF
```

Step II: Activate the Request

1. Identify the queue overrides

There is no need to define any queue overrides.

2. Queue the request:

```
QUEUE  #REQLJLF
```

Procedures for Remote Job Submission

This section discusses the BCOM JOB submission facility as it pertains to jobs executing on other platforms.

BCOM actually creates on NSK Guardian, a TACL or COMINT process and passes the user's Obey file as the IN file and what the user specifies in the BCOM OUTFILE command as the OUT.

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

> ***Note:*** *NSK Guardian supports WAITED job submissions. You can submit jobs and then wait for their completion.*

Remote Job - Local File (MVS to NSK Guardian)

A request that is defined as a remote job will execute on the remote platform is identified by the REMOTE-LOCATION keyword.

When a job file resides locally, the job data is transferred before the job is submitted.

Step I: Define the Request

An Obey file submission 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 Obey file submission request you want to execute is already defined, you do not have to go through Step I.

1. Get ready

In order to define an Obey file submission request, the following information must be known:

About the Obey file: The Obey file can reside either on the Tandem or another IBM/MVS disc. Use the appropriate LOCAL-FILE or REMOTE-FILE descriptor.

On NSK Guardian, the Obey file can be a disc file or a TANDEM DNS ALIAS name. For a disc file, the complete file name must be known. For a DNS ALIAS name, the name must be present in the distributed database of names. The Obey file or DNS ALIAS does not need to exist at this point.

On MVS, the dataset, which contains the appropriate commands, can be a disc file. The complete file name must be known.

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

MVS Information:

LOCAL-LOCATION, LOCAL-FILE and LOCAL-ATTRnn keywords are used to define JOB input data being sent from MVS to a remote NSK Guardian location.

NSK Guardian Information:

REMOTE-LOCATION, REMOTE-FILE and REMOTE-ATTRnn keywords are used to define OBEY data that is to be submitted at the location defined as the REMOTE-LOCATION.

General Information:

Request context: BCOM allows you to specify the next request name to be executed upon the completion of this Obey file submission. The next request can be of any type (SF, RF, LJ, RJ) and a different one can be chosen depending on the outcome of the ancestor request. The chained request may be selected based on the completion status of the Obey request (normal or abnormal).

Note that the completion status of an Obey file submission request only relates to the submission itself; the normal or abnormal end of any step included in this Obey file is not handled by BCOM.

2. Set the Request Attributes

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

General Information:

```
SET  TYPE             RJ
SET  LOCAL-LOCATION   logical-location-name
SET  PRIORITY-CLASS   priority-class-value
```

MVS Information:

```
SET  LOCAL-ATTR01     file-disposition
SET  LOCAL-FILE       MVS-file-name
```

NSK Guardian Information:

```
SET  REMOTE-LOCATION  logical-location-name
SET  ROUTING-CLASS    routing-class-value
SET  REMOTE-FILE      Guardian-file-name
SET  REMOTE-ATTR01    OUTFILE=outfile-name
SET  REMOTE-ATTR02    process-priority=100
SET  REMOTE-ATTR03    JOB-WAITED=yes
```

3. Add the request

Immediately following the SET commands (within the same BINT 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 BINT session or at a later time. The Obey file must be present on the appropriate node.

1. Identify the queue overrides

This step is not required for an Obey file submission request where LOCAL-LOCATION and REMOTE-LOCATION are the same.

2. 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, with the same process-priority will be processed first.

Example 1 - Remote Job with Local File

Submit a Remote Job with NSK Guardian Obey commands that reside on MVS.

Step I: Define the Request

1. Get ready

- The Obey file name is MVS-OBEY-FILE
- The OUTFILE name to be used is $S.#OBEYOUT

2. Set the request attributes

```
READY
SET  TYPE              RJ
SET  LOCAL-LOCATION    %MVS1
SET  LOCAL-FILE        MVS-OBEY-FILE
SET  LOCAL-ATTR01      DISP=SHR
SET  REMOTE-LOCATION   %TAND1
SET  REMOTE-ATTR01     OUTFILE=$S.#OBEYOUT
SET  REMOTE-ATTR02     PROCESS-PRIORITY=100
SET  REMOTE-ATTR03     JOB-WAITED=YES
SET  PRIORITY-CLASS    03
SET  ROUTING-CLASS     01
```

3. Add the request

```
ADD  #REQOBY1
```

Step II: Activate the Request

1. Identify the queue overrides

There is no need to define any queue overrides.

2. Queue the request

```
QUEUE #REQOBY1
```

Example 2 - Remote Job with Remote File

Submit a Remote Job with NSK Guardian Obey commands, which reside on NSK Guardian.

Step I: Define the Request

1. Get ready

- The Obey file name is $DATA01.MYOBEY.OBEY
- The OUTFILE name to be used is $S.#OBEYOUT

2. Set the request attributes

```
READY
CALLBINT
SET  TYPE              RJ
SET  LOCAL-LOCATION    %MVS1
SET  REMOTE-LOCATION   %TAND1
SET  REMOTE-FILE       $DATA.MODIFY.OBEY
SET  REMOTE-ATTR01     OUTFILE=$S.#OBEYOUT
SET  PRIORITY-CLASS    03
SET  ROUTING-CLASS     01
```

3. Add the request

```
ADD  #REQOBY2
```

Step II: Activate the Request

1. Identify the queue overrides

There is no need to define any queue overrides.

2. Queue the request

```
QUEUE  #REQOBY2
```

Report Printing

This section discusses the BCOM Report Printing Facility. It illustrates the basic categories of report distribution requests supported by the product and provides several examples of how to set up, effect and control such transfers.

The section is organized according to the following sub-sections:

1. What is the Report Printing Facility? gives a general description of this feature. It illustrates the architecture and explains how the facility works.

2. Report Distribution explains how to use the Print feature for automatic report printing. Examples of how to print reports from MVS to a remote platform are included, as well as a brief discussion of printing from a remote platform to MVS.

What is the Remote Printing Facility?

In this subsection we examine how BCOM can be used to route or distribute reports between platforms. The report printing facility does not require the definition of a request to BCOM. The transfer is initiated through the platform's spool subsystem (JES on MVS).

The remote printing facility for the MVS platform consists of an external writer/output writer combination. The external writer runs as a started task. The output writer runs as a sub-task of the external writer.

Report Printing Architecture: MVS to a remote platform

On the MVS partner, B62XTWR uses the Output Writer to communicate with B62MAIN, which is responsible for the data transfer. Its partner on the remote platform is the module BFSSER (server).

Interaction between External Writer and JES Spooler

The external writer interacts with the JES by requesting print data in one of two ways:

- Select data by class or a set of classes
- Select data by a destination

Whether selecting by class or by destination, whenever output is generated for that sysout class or destination, JES notifies the external writer.

What the external writer does

When the external writer is notified by JES, it does the following:

- Receives the information necessary to access the print data
- Compares the destination associated with the print data (whether selected by class or destination)
- Searches for a match in the configuration table
- Sends routing attributes to the output writer

> ***Note:*** *If a match is made, the corresponding routing attributes are transmitted. If no match is made, the default routing attributes are passed on.*

What the Output Writer Does

- Receives the routing attributes
- Accesses the print data and transfers it to the correct main task

Transfer to Server Platform

The main task transfers the print data to the correct server platform, based on the routing attributes defined in the configuration table.

Report Distribution (Spool to Spool)

To Print Reports From MVS to a Remote Platform

The three examples below illustrate the various ways to print reports from MVS to a remote platform. In all the examples, we assume that the following configuration exists on MVS:

- The B62XTWR job has been started with class C in the parameter string.
- The default Output Writer for class C is B62OUT.
- There is another BJES Output Writer called B62OUTC.

Example 1: *In this example:*

- no specific Output Writer is specified, so the default (B62OUT) will be used
- the report will be sent to process-id defined for B62OUT
- the form name is STD and the number of copies is 1

```
//STEP1    EXEC PGM=IEBGENER
//SYSUT1   DD   DSN=.........,DISP=SHR
//SYSUT2   DD   SYSOUT=C
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
```

Example 2: *In this example:*

- the B62OUTC Output Writer is specified
- the report will be sent to the process-id defined for B62OUTC
- the form name is STD and the number of copies is 1

```
//STEP1    EXEC PGM=IEBGENER
//SYSUT1   DD   DSN=.........,DISP=SHR
//SYSUT2   DD   SYSOUT=(C,B62OUTC)
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
```

Note: *If a non-default output writer is specified in the SYSOUT card, the BJES EXTERNAL WRITER will require an approval at the MVS operator console.*

Example 3: *In this example:*

- the B62OUTC Output Writer is specified
- the number of copies is 44
- the form number is 997

```
//STEP1    EXEC PGM=IEBGENER
//SYSUT1   DD   DSN=.........,DISP=SHR
//SYSUT2   DD   SYSOUT=(C,B62OUTC,997), DEST=REMOTE22,COPIES=44
//SYSPRINT DD   SYSOUT=*
//SYSIN    DD   DUMMY
```

Note: *If a non-default output writer is specified in the SYSOUT card, the "BJES EXTERNAL WRITER" will require an approval at the MVS operator console.*

To Print Reports From a Remote Platform To MVS

When transferring reports from a remote platform to be printed on the MVS, be sure that if the print transfer is defined by a class or form, that same class or form already exists on the MVS side.

BCOM Follow-on Scheduling

This section discusses how to request the Follow-on Scheduling, a BCOM facility 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, which illustrates the flexibility and ease of use of this powerful facility.

The section is organized according to the following sub-sections:

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.

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

When the Automatic Scheduling function is used in conjunction with BCOM 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 being normal or abnormal termination.

When a BCOM request is defined via the BINT, 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.

The completion information is based on the termination status of the process or the task that carries out the transfer request. For example, in the case of a job submission to NSK Guardian, the termination status is either a STOP or an ABEND message generated when the attached TACL or COMINT process has terminated.

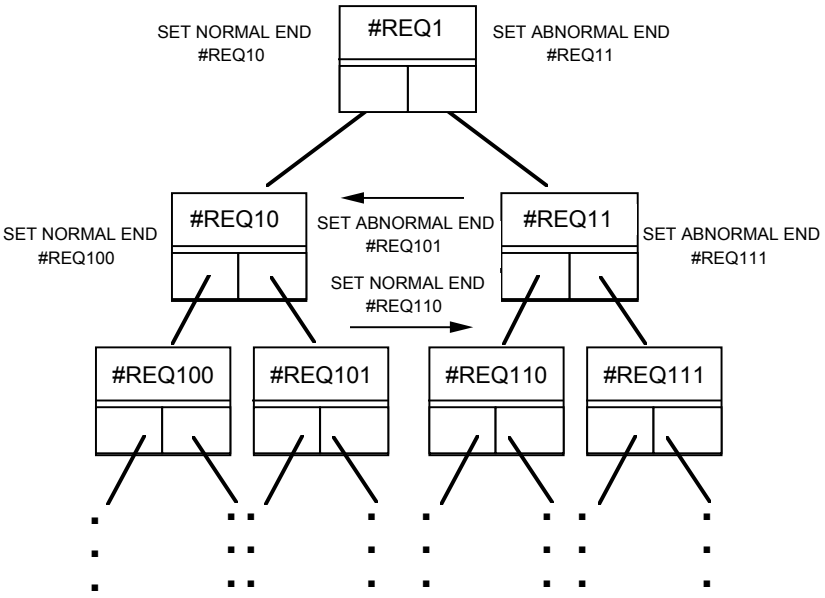
Request Definition Overview

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 run-away 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.

Figure 11-1



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 BINT program.

For example, a job submitted to a remote platform may include a step in which the local BINT - 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.

Remote Queueing 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 on 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 sub-section discusses how to prepare for and use the Remote Queuing facility of BCOM.

Local and Remote Queueing

All BCOM transfer activities, except print distribution, must be pre-defined and stored as request definitions in a repository file called the Request File. To initiate these requests, 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 BCOM remote queuing facility 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 or ABNORMAL-END keywords of a request definition. This will implicitly schedule the remote queue request based on the completion status of the ancestor request.

The Remote Queue Request

A Remote Queue request is like any other request definition in the request file of your local BCOM location except that its remote file name is actually the name of a request that exists on the request's remote location.

However, since you specify the remote request name in the local request - under the same or a different name - you can, for example locally maintain a record of what is queued remotely. In addition, your remote queue request is always under control, because you can keep track of its scheduling and progress, using the Queue management commands of BINT and the BCOM logs at both ends provide you with a detailed trace of its execution.

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 pre-defined BCOM request, the Remote Queue request uses the remote-filename keyword to specify the desired request name.

Here is a sample definition:

```
SET  TYPE             RQ
SET  LOCAL-LOCATION   %MVS1
SET  REMOTE-LOCATION  %TAN1
SET  REMOTE-FILE      #EFGH
SET  ROUTING-CLASS    01
SET  PRIORITY-CLASS   05
SET  COMMENT          queue request #EFGH at %TAN1
ADD  #ABCD
```

The above sample request will be queued at priority 05 to the routing class 01 of the local BCOM on %MVS1. When it runs, the request will cause that local BCOM to send a remote queue command to the BCOM running at remote location %TAN1. This remote queue command will initiate request #EFGH, found in the request file on %TAN1.

Alternate Way of Defining the Request Name

There is an alternate way of providing the name of the request to be scheduled on 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 on 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 IBM/MVS corporate mainframe to a set of different BCOM locations, using the Remote Queuing Facility.

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: on MVS, BCOM will be responsible for distributing the corporate file to its adjacent nodes, and the BCOM applications 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; its NORMAL-END then triggers the local request #SNDFVMS

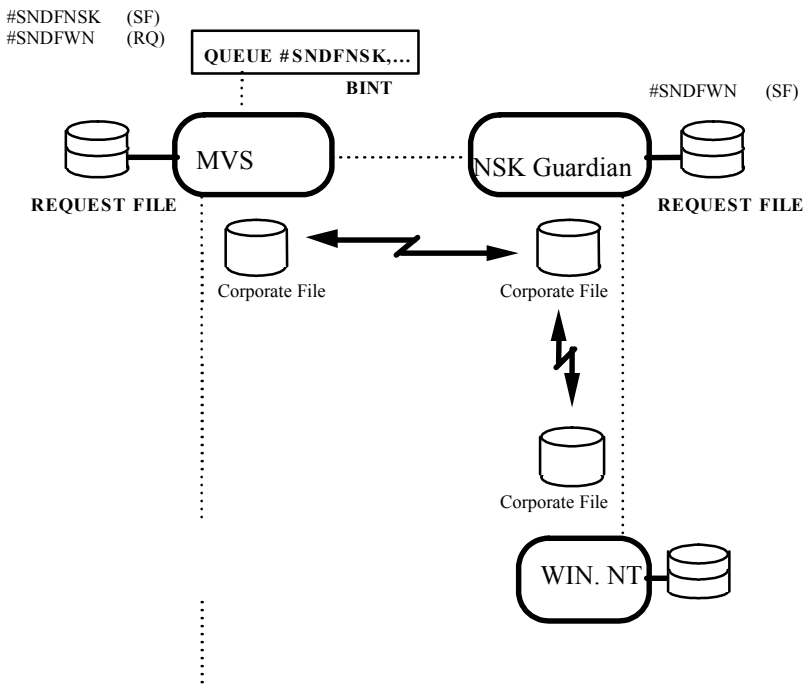
The NSK Guardian Request File

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

- #SNDFWN - a SEND-FILE to distribute the 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)
```

Local Platform Security

The ID of the user who 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 activated 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 of 10), 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 on that remote BCOM location
- remote request specified is already scheduled or queued for execution on that remote BCOM location
- remote request was queued successfully

Other Remote Queueing 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.

Time Scheduling Facility (TSF)

BCOM can facilitate work management across multiple platforms, thanks to a number of built-in features that include the Remote Queuing Facility (RQF) and the Timer Scheduling Facility (TSF).

The purpose of this sub-section is to describe BCOM's on-board Timer Scheduling Facility and show you how to schedule requests for later execution, without the need for an external scheduler software.

Using the Time Scheduling Facility

With the Time Scheduling Facility (TSF) of BCOM, you can:

- set a specific execution date and time for your requests;
- ask that your requests be automatically re-scheduled by BCOM at fixed, repetitive intervals of days;
- ask that your requests be automatically re-scheduled by BCOM at fixed, repetitive intervals of hours.

This information is specified as additional parameters on the QUEUE command - it does not apply to QWAIT.

BCOM must be up and running before you can queue a request that specifies Time Scheduling Facility parameters. However, once such requests have been queued, BCOM will keep (even across restarts) monitoring the queue file for TSF entries until their scheduled time have arrived.

> *Note: When using the Time Scheduling Facility, your are not allowed to queue more than one request at a time within the same command line.*

Command Syntax

A scheduling-date-time and a scheduling-interval appear as the third and fourth optional parameters of the QUEUE command.

An example of the Queue command syntax with TSF keywords follows:

```
QUEUE    <#request-name> ( [ <Routing-Class> ]
         [ , <Priority-Class> ]
         [ , <scheduling-date-time> ]
         [ , <scheduling-interval> ]    )
```

where:

<*scheduling-date-time*>

Define the date and the time at which the request should be executed;

It takes the form of a fully qualified timestamp - such as <YYYYMMDDHHMM>:

YYYY	year	(current year-2099)
MM	month	(01-12)
DD	day	(01-31)
HH	hour	(00-23)
MM	minute	(00-59)

<*scheduling-interval*>

Define the repetitive interval at which that request will be automatically re-scheduled for execution.

You have a choice of specifying a daily interval or an hourly interval:

ddd defines the number of days interval;

Hhh defines the number of hours: the first character being mandatory character 'H' and the two others representing a number between 1 and 23.

Examples of How to Use the TSF

Here are a few examples of how to use the BCOM Time Scheduling Facility (TSF).

Example 1:

Execute a request once, on December 31st, 1998, at 9:30 AM.

- assume the request is called #VWXYZ
- December 31st, 1998 is 1998-12-31; 9:30 AM is 09:30 hours;
- there is no repetition factor to supply:

```
QUEUE  #VWXYZ(,,1998-12-31 09:30)
```

Example 2:

Execute a request on every Wednesday, starting with September 09, 1998, at 8 PM.

- assume the request is called #ABCD;
- September 09, 1998 is 1998-09-09; 8 PM is 20:00 hours;
- specify 007 to repeat the scheduling every week:

```
QUEUE  #ABCD (,,1998-09-09 20:00,007)
```

Example 3:

Execute a request every 12 hours, starting at 10 AM on September 20th, 1998, where the priority class and routing class are overridden by 05 and 01 respectively.

- assume the request is called #ASDF;
- September 20th, 1998 is 1998-09-20; 10 AM is 10:00 hours;
- specify H12 to repeat the scheduling every 12 hours;
- assume the remote queue is sent using a routing class of 01 and pri-class 05:

```
QUEUE  #ASDF(01,05,1998-09-20 10:00,H12)
```

Considerations and Restrictions

- Time Scheduling Facility (TSF) is offered on the MVS platform only.
- If the starting date and time is already past, then your QUEUE command will be rejected.

BCOM Error-Handling Facilities

The purpose of this section is to discuss various BCOM facilities 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 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 inter-connections.

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.

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 retry counter has not expired, 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 us see how you can make use of this information.

Suppose the majority of your operationsthat can be retried come from establishing sessions with a remote BCOM that 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 AUTO-START 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. For example. if BCOM has just completed another unsuccessful attempt at re-establishing the connection to your remote location when you are informed that the communication resources formerly 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 a 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 multi-thread 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 singleplex 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 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, an 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 Tandem's SOX and an external X25PAD on MVS.

When you initiate a request (with the Queue command), you can even override a pre-defined 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 automatically restarts 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    02
SET  AUTO-RESTART     YES
...
```

If you display the items in the queue while your request is to be re-started, you will see that BCOM assigns it a special status of "waiting for auto-restart".

Impact of re-execution

The MVS file disposition (JCL parameter DISP=) can be an inconvenient when a transfer to BCOM MVS 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 in the filename at restart time, avoiding manual modification of the request to restart.

Restrictions:

- DISP=SCR, DISP=MOD are not supported
- to create a file on the 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 cataloged, 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)”

Checkpoint-Restart

CHECKPOINT-RESTART is a different feature, designed to recover from long-running data transfers. It is a request definition facility similar to the AUTO-RESTART, and BCOM will keep the failed request - for whatever the reason - into the queue file until you do a manual restart.

The distinctive feature of CHECKPOINT-RESTART is that it allows you to select a checkpoint interval at which the BCOM programs on each side of the link can exchange their current read and write positions as they process their respective dataset. This checkpoint information allows them to pick up where they have left, once you have re-started the request.

Here is how you specify this facility in your request definition; note that the default value for the CHECKPOINT-RESTART keyword is "000" (meaning 'none').

```
SET  TYPE              SF
SET ROUTING-CLASS      02
SET CHECKPOINT         nnn
•••
```

The value specified for checkpoint interval (nnn) represents a record counter:

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

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

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

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

Impact of re-execution

The MVS file disposition (JCL parameter DISP=) can be an inconvenient when a transfer to BCOM MVS 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 in the filename at restart time, avoiding manual modification of the request to restart.

Restrictions:

- DISP=SCR, DISP=MOD are not supported
- to create a file on the 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 cataloged, 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. Thus it can become a key element of your operations strategy, especially when pressed for smaller windows of transfer time.

There are certain considerations for using the CHECKPOINT-RESTART facility:

- this facility is available only for structured files - e.g. not for NSK Guardian unstructured files;
- you must manually re-start the transfer, using the QRESTART command;
- the QRESTART command allows you to override the checkpoint value and specify a restart "from the beginning". The MVS disposition and generation +1 are still interpreted as in other restart cases.

Abnormal-End

ABNORMAL-END is another recovery facility, designed this time to handle requests that complete abnormally. It is a request definition facility described as 'Follow-on Scheduling': it represents, in effect, an implicit method of queuing additional requests into BCOM.

If either the local or the remote BCOM program detects an error in the processing of the request (such as security violation, file not found, etc.), then the location that initiated the transfer will terminate the request abnormally; if an ABNORMAL-END is specified in the request, then BCOM will automatically schedule that next request for you. This, of course, also applies to NORMAL-END cases.

Here is how you specify this facility in your request definition; note that by default, requests are not chained in the request file: you must explicitly reference another request by name in order to establish a chain of requests.

```
SET  TYPE              SF
SET ROUTING-CLASS      02
•••
SET ABNORMAL-END       #NEXTRQ
•••
```

These keywords are optional; and you can also specify only one of them.

Local and Remote Queueing

Both keywords of follow-on scheduling accept the name of a local or a remote request; for example:

```
SET  TYPE              SF
SET LOCAL-LOCATION     %MVS1
•••
SET REMOTE-LOCATION    %TAN1
SET ROUTING-CLASS      02
•••
SET ABNORMAL-END       #ABCD
```

will cause the initiating BCOM at %MVS1 to schedule the request #ABCD from its local request file.

By contrast, the following request definition:

```
SET  TYPE              SF
SET LOCAL-LOCATION     %MVS1
...
SET REMOTE-LOCATION    %TAN1
SET ROUTING-CLASS      02
...
SET ABNORMAL-END       #EFGH
```

will cause the same initiating BCOM at %MVS1 to send a remote queue command to the BCOM running at the remote-location %TAN1 to schedule the request #EFGH from that remote platform's Request file.

Considerations

Auto-Restart and Check-Point Restart

AUTO-RESTART and CHECKPOINTING can be used together. If a request fails on an "auto-restartable" condition, the restart will occur automatically from the last checkpoint. If the condition is not "auto-restartable", it will be in the Queue file in a manual restart condition.

Auto-Restart and Abnormal-End

Note that BCOM always triggers the ABNORMAL-END (if defined) request when a transfer failure occurs for an AUTO-RESTART request; this trigger can be used to warn the operator, schedule a special job, etc.

Appendix A - Command Syntax and Reference Summary

This appendix provides an alphabetical list of the user interface commands that can be used with BCOM for MVS.

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, either upper or 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. Punctuation must be entered exactly as shown

7. Request names begin by the character #.

Process names begin by the character $.

Location names begin by the character %.

ABORT

Function:

To force a transfer process to abend, 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.

Syntax:

The basic syntax for the command is as follows:

```
ABORT  #request | transfer process | ALL
```

When ABORTing a request, use the following syntax:

```
ABORT #request-name [, #request-name, #request-name, ...]
```

When ABORTing a transfer process, use the following syntax:

```
ABORT $transfer-process-name
```

Typical uses

When aborting one request, code

```
ABORT #REQABC
```

When aborting more than one request, code

```
ABORT #REQ1, #REQ2, #REQ3, ...
```

When aborting a single transfer process, code

```
ABORT $BFS62
```

When aborting all the transfer processes within all the routing classes servicing all the remote locations defined in the local Configuration file, code

```
ABORT ALL
```

Considerations

This command can be used to interrupt one specific file transfer. When a Send File is in progress and the transfer process is aborted, the file that 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 command.

Example 1:

The following ABORT command forces the $EXAMP process to abort.

```
ABORT $EXAMP
PROCESS $EXAMP HAS BEEN ABORTED
```

Example 2:

The following ABORT command forces the #C1O006 request to abort.

```
ABORT #C1O006
REQUEST #C1O006 HAS BEEN ABORTED
```

ADD

Function:

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

Syntax:

```
ADD  #request-name
```

Considerations:

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.

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
```

COMMENT

Function:

To support a comment line within a command file.

Syntax:

```
COMMENTn [comment-text]
```

Where

n 1 to 5

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. A maximum of 5 contiguous comment-text lines is supported.

Example:

Simple use of the COMMENT command.

```
COMMENT  ****  THIS IS NOT PROCESSED BY BINT  ****
```

Note: *BCOM also supports request comments, which can be entered via the SET COMMENTnn command, and are stored on the Request file.*

DRAIN

Function:

To stop a transfer process within a routing class within a remote location after completion of any transfers that are currently in progress.

Syntax:

```
DRAIN  $transfer-process-name | ALL
```

Typical uses

When draining a single transfer process, code:

```
DRAIN  $BFS62
```

When draining all the transfer processes within all the routing classes servicing all the remote locations defined in the local Configuration file, code:

```
DRAIN  ALL
```

Considerations:

The DRAIN command is what is normally used to stop a transfer process. The transfer process will terminate in an orderly fashion, and only after the current active file transfer(s) have completed normally.

Example:

The following command drains the $EXAMP process.

```
DRAIN $EXAMP
DRAIN IS IN PROGRESS

STATUS $EXAMP
```

REMOTE LOCATION	RTE CLASS	RTE TYPE	PROCESS ID	JOBS SELECT	STATUS	CONNECTED	MAX RQST	ENABLE COMPRS
%TAND1	01	STANDARD	$EXAMP	01	DRAINING	YES	1	NO
%TAND1	02	STANDARD	$EXAMP	02	DRAINING	YES	2	NO

LOGICAL UNIT NAME	SESSION ID	REQUEST NAME	NB BLOCKS TRANSFERRED	STATUS	ERROR CODE	ERROR REASON	RET	FBK
LU1	0224	#$Q01A	0000042	DRAINING	0000	0000	00	00
LU2	0225	#RQ01B	0000012	DRAINING	0000	0000	00	00
LU3	0226	#RQ01C	0000064	DRAINING	0000	0000	00	00

EXIT

Function:

To terminate the execution of the User Interface module.

Syntax:

```
EXIT
```

Considerations:

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

Example:

This command terminates execution of BINT, regardless of the mode BINT was invoked with.

```
EXIT
READY
```

HELP

Function:

This command displays help information for a particular command or for all the commands.

Syntax:

```
HELP  [command | ALL]
```

where

command identifies a BINT command

ALL specifies all BINT commands and is the default.

Considerations:

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
```

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 or ALL which will list the attributes of all requests.

Example:

The following command displays file transfer attributes for the #DUMP2 request.

```
INFO  #DUMP2
REQUEST = #DUMP2
TYPE = SEND FILE
RQST-COMMENTS:
01 This is a transfer request of a currently existing
02 file.
ROUTING-CLASS = 02
PRIORITY-CLASS = 2
LOCAL-LOCATION = %MVSMTL
LOCAL-FILE = MVS.TEST.DSN
LOCAL-FILE-ATTRIBUTES:
01  DISP=SHR
REMOTE-LOCATION = %TAND1
REMOTE-FILE = $TEST.FILE
REMOTE-FILE-ATTRIBUTES:
01  PROCESS-PRIORITY  170
02  REPORT OFF
AUTO-RESTART = NO
CHECKPOINT   = NO
COMPRESSION  = NO
ABNORMAL-END = NONE
NORMAL-END   = NONE
OWNER ID     = $OWNER-ID
FIELD   04,02  10, 04
```

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.

Considerations:

If you omit the parameters 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  FILE  #EX
#EXAMP10  #EXAMP21 #EXAMP52 #EXCHG #EXTRA
```

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.

Considerations:

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

There are two possible syntaxes for the Queue command, depending on whether you are sending one or more requests at a time. These are:

Case 1: for ONE request

```
QUEUE <#request-name>
      ([ <Routing-Class> ]
       [ , <Priority-Class> ]
       [ , <scheduling-date-time> ]
       [ , <scheduling-interval> ])
```

where:

#request-name is the file transfer request to be queued,

routing-class is the routing class over which this request will be transferred,

priority-class represents the priority which with this request will be queued.

<scheduling-date-time>
defines the date and the time at which the request should be executed. It takes the form of a fully-qualified timestamp-such as

`<YYYY-MM-DD-HH:MM>,`

where:

YYYY	year	(current year-2099)
MM	month	(01-12)
DD	day	(01-31)
HH	hour	(00-23)
MM	minute	(00-59)

<scheduling-interval>
defines the repetitive interval at which that request will be automatically re-scheduled for execution.

You have a choice of specifying a daily interval or an hourly interval:

ddd	defines the number of day interval;
Hhh	defines the number of hours.

Must be prefixed by the letter "H".

Case 2: for MORE THAN ONE request

```
QUEUE  <#request-name> [,#Request-name ...]
```

You have already noticed that the optional parameters are not allowed when sending many requests at a time.

Example 1:

The following command queues request #EXAMP1 for file transfer.

```
QUEUE #EXAMP1
#EXAMP1 QUEUED
```

Example 2:

The following command queues request #EXAMP1 for file transfer and specifies a routing-class 01, a priority-class 02, a scheduling-date-time which represents November 15th, 1995, at 22:30 hrs and a scheduling-interval of six hours.

```
QUEUE #EXAMP1(01,02,1995-11-15 22:30,H06)
#EXAMP1 QUEUED
```

Example 3:

The following command queues requests #EXAMP2, #EXAMP3 and #EXAMP4 for file transfer.

```
QUEUE #EXAMP2,#EXAMP3,#EXAMP4
#EXAMP2 QUEUED
#EXAMP3 QUEUED
#EXAMP4 QUEUED


QLIST
```

REQUEST	PRI	QUEUE TIME	STATUS	RLOC	RTECLAS	CHKPNT	RSTRT
#EXAMP1	01	1998-10-01 10:15:06	EXECUTING	%TAND1	01	No	No
#EXAMP2	02	1998-10-01 10:01:20	EXECUTING	%TAND2	02	500	No
#EXAMP3	05	1998-10-01 10:18:19	SELECTED	%TAND3	05	No	No
#EXAMP4	09	1998-09-30 12:05:06	RETRY STATE	%TAND4	03	No	No

QUEUEALT-PRI | QALT-PRI

Function:

Alter the priority-class of a queued request.

Syntax:

```
QALT-PRI  #request-name, pri-class-value
```

where:

request-name is the file transfer request whose priority-class is to be altered.

pri-class-value is the new value for the priority class to be given to the request.

Considerations:

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.

```
QALT-PRI  #EXAMP,9
THE PRIORITY HAS BEEN ALTERED
```

QUEUEDEL

Function:

Delete a queued request from the Queue file if the transfer has not yet started.

Syntax:

```
QDEL  #request-name [, #request-name, #request-name...]
```

where:

request-name is the file transfer request to be deleted.

Considerations:

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

Function:

Hold a queued request from being executed before it has begun. The request will not lose its FIFO order within the queue.

Syntax:

```
QHOLD  #request-name [, #request-name, #request-name...]
```

where:

request-name is the file transfer request to be held.

Considerations:

The request(s) to be held must be currently queued, but not executing.

Example 1:

This command tries to hold the queued request #EXAMP2.

```
QHOLD  #EXAMP2
#EXAMP2  NOT HELD, CURRENTLY EXECUTING
```

QUEUELIST

Function:

Display the currently queued requests.

Syntax:

```
QLIST
```

Considerations:

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	QUEUE TIME	STATUS	RLOC	RTECLAS	CHKPNT	RSTRT
#EXAMP1	01	1998-10-01 10:15:06	EXECUTING	%TAND1	01	No	No
#EXAMP2	02	1998-10-01 10:01:20	EXECUTING	%TAND2	02	500	No
#EXAMP3	05	1998-10-01 10:18:19	SELECTED	%TAND3	05	No	No
#EXAMP4	09	1998-09-30 12:05:06	RETRY STATE	%TAND4	03	No	No

QUEUEREL

Function:

Release a previously queued request which has been held with the QUEUEHOLD command.

Syntax:

```
QREL  #request-name [, #request-name, #request-name...]
```

where:

#request-name is the file transfer request to be released.

Considerations:

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	QUEUE TIME	STATUS	RLOC	RTECLAS	CHKPNT	RSTRT
#EXAMP1	01	1998-10-01 10:15:06	HELD	%TAND1	01	No	No
#EXAMP2	02	1998-10-01 10:01:20	EXECUTING	%TAND2	02	500	No
#EXAMP3	05	1998-10-01 10:18:19	SELECTED	%TAND3	05	No	No
#EXAMP4	09	1998-09-30 12:05:06	RETRY STATE	%TAND4	03	No	No

```
QUEREL # EXAMP1
REQUEST #EXAMP1 HAS BEEN RELEASED
QLIST
```

REQUEST	PRI	QUEUE TIME	STATUS	RLOC	RTECLAS	CHKPNT	RSTRT
#EXAMP1	01	1998-10-01 10:15:06	EXECUTING	%TAND1	01	No	No
#EXAMP2	02	1998-10-01 10:01:20	EXECUTING	%TAND2	02	500	No
#EXAMP3	05	1998-10-01 10:18:19	SELECTED	%TAND3	05	No	No
#EXAMP4	09	1998-09-30 12:05:06	RETRY STATE	%TAND4	03	No	No

QUEUERESTART

Function:

To release a previously queued request which ended abnormally and is in state HELD for RESTART.

Syntax:

```
QUEUERESTART #request-name [,#request-name ...] [COLD | C]
```

where:

#request-name is the file transfer request to be re-started.

COLD Re-start from the beginning.

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 operator's 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.

Example:

This example restarts a previously held request, using the current checkpointing information:

```
QUEUERESTART  #EXAMP1
#EXAMP1 FILE TRANSFER HAS BEEN INITIATED
#EXAMP1 HAS COMPLETED SUCCESSFULLY
```

QUEUEWAIT

Function:

To initiate a file transfer request and wait for its completion.

Syntax:

```
QUEUEWAIT  #request-name [routing-class, priority-class]
```

Considerations:

When QUEUEWAIT is issued, the BINT session is suspended until the file transfer is completed.

If BINT 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 BINT will be ready for another command.

When running in batch mode, the status of the request is displayed. If a failure is detected, BINT will set the condition code.

If BINT is run in single command mode, the status of the request will be displayed. If a failure is detected, BINT will set the condition code.

The optional parameters include queue override options (except scheduling). 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.

A comma placeholder must be entered if the routing class is omitted and the user wishes to override the priority-class. 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
#EXAMP1  FILE TRANSFER HAS BEEN INITIATED
#EXAMP1  HAS COMPLETED SUCCESSFULLY

STATUS #EXAMP1
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
```

Example 2:

This command initiates the file transfer request #EXAMP2 to be transferred over routing class 02, 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.

```
QWAIT #EXAMP2 (02)
#EXAMP2  FILE TRANSFER HAS BEEN INITIATED
#EXAMP2 HAS NOT COMPLETED SUCCESSFULLY, JCL ERROR

STATUS EXAMP2
REQUEST NAME = #EXAMP2
QUEUE TIME = 1998-03-31 13:14:55
START TIME = 1998-03-31 13:14:55
END TIME = 1998-03-31 13:14:58
RECORD COUNT = 0
BLOCK COUNT = 0
MESSAGE = JCL ERROR, DELETED
```

RESET

Function:

To re-initialize with spaces or default values all or a specific parameter prior to creating a request, to starting a transfer, or to queuing a request.

Syntax:

```
RESET  [reset-option | ALL]
```

where:

reset-option can be any of the following:

```
ABNORMAL-END
AUTO-RESTART
CHECKPOINT
COMPRESSION
FIELD
LOCAL-ATTRnn
LOCAL-FILE
LOCAL-LOCATION
NORMAL-END
PRIORITY-CLASS
ROUTING-CLASS
COMMENTnn
TYPE
REMOTE-LOCATION
REMOTE-ATTRnn
REMOTE-FILE
```

ALL refers to all of the options.

Example:

The following RESET command will reset the value of the REPORT field to OFF.

```
SET LIKE #DUMP2
SHOW
REQUEST PARAMETERS:
RQST-TYPE = SEND FILE
RQST-COMMENTS:
01 This is a transfer request of a currently
02 existing file
ROUTING-CLASS = 02
PRIORITY-CLASS = 2
LOCAL-LOCATION =%MVSMTL
LOCAL-FILE = MVS.TARGET.DSN
LOCAL-FILE-ATTRIBUTES:
01  DISP=SHR
REMOTE-LOCATION = % TAND1
REMOTE-FILE =$DATA01.FILE.TEST
REMOTE-FILE-ATTRIBUTES:
01  PROCESS-PRIORITY  170
02  REPORT ON
AUTO-RESTART = NO
CHECKPOINT   = NO
COMPRESSION  = NO
NORMAL-END   = NONE
ABNORMAL-END = NONE
FIELD = 0,48 52,2 58,4

RESET REMOTE-ATTR02

SHOW
REQUEST PARAMETERS:
REQUEST-TYPE = SEND-FILE
RQST-COMMENTS:
  01  THIS IS A TRANSFER REQUEST OF A CURRENTLY
  02  EXISTING FILE
ROUTING-CLASS = 02
PRIORITY-CLASS = 2
LOCAL-LOCATION = %MVSMTL
LOCAL-FILE = MVS.TARGET.DSN
LOCAL-FILE-ATTRIBUTES:
  01  DISP = SHR
REMOTE-LOCATION = %TAND1
REMOTE-FILE = $DATA01.FILE.TEST
REMOTE-FILE-ATTRIBUTES:
  01  PROCESS-PRIORITY 170
AUTO-RESTART = NO
CHECKPOINT   = NO
COMPRESSION  = NO
NORMAL-END   = NONE
ABNORMAL-END = NONE
FIELD = 0,48 52,2 58,4
```

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                  OR
SET  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:

```
NORMAL-END
ABNORMAL-END
AUTO-RESTART
CHECKPOINT
COMPRESSION
COMMENT
FIELD
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:

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-QUEUE)** 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 use to specify the name of a remote request to be queued at a remote location.

RB **(REMOTE-BACKUP)** indicates a request for executing a backup request on the remote location node.

RR **(REMOTE-RESTORE)** indicates a request for executing a restore request on the remote location.

SP (SEND-PDS - For transfers between MVS's only) specifies a PDS to PDS file transfer from the local location to the remote location.

RP (RECEIVE-PDS - For transfers between MVS's only) specifies a PDS to PDS file transfer from the remote location to the local 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 BINT program that processes one command. The name must exist in the Configuration file. Location names begin with the % sign.

`LOCAL-FILE dataset-name`

Identify the local dataset name.

`LOCAL-ATTRnn | LATTRnn identifier=value`

is used to define specific attributes for the LOCAL file and location.

Up to 10 local attributes may be entered:

`STARTFILE=Guardian-filename`

This attribute is valid only for restore requests executed by BCOM MVS acting as a server, for a requester in BCOM on NSK-Guardian. This attribute can not be set in a RR request defined in BCOM on MVS.

STARTFILE minimizes the amount of data transmitted for a restore request. When specified, BCOM on MVS does not transmit the Guardian files included in the backup on MVS, until the specified Guradian-filename is found.

Only two formats are supported: “\node.$volume.subvol.file” and “$volume.subvol.file”. If the node name can be specified, it is always ignored.

If the specified file is not found in the MVS file, the request is aborted.

`DCB=(DSN,LRECL=xxxx,BLKSIZE=xxxx,DEN=n,DSORG=aa,RECFM=xxxx)`

Specify the Data-Control-Block of the the MVS dataset. Only the following DCB parameters are accepted by BCOM: LERCL, BLKSIZE, DEN, DSORG RECFM, and dataset name which refers to GDG model.

DSN	Reference to a dataset name for its attributes.		
LRECL	5 numeric digit maximum		
BLKSIZE	5 numeric digit maximum		
DEN	2	(800bpi)	or
	3	(1600bpi)	or
	4	(6250bpi)	
DSORG	PS	(Physical Sequential)	or
	PO	(Partitionel Organization)	
RECFM	F	(fixed)	*or*
	V	(variable)	or
	FB	(fixed, blocked)	or
	VB	(variable, bloacked)	or
	U	(unstructured)	or
	VS	(variable, spanned)	

`DISP=(status,normal-disp,abnormal-disp)`

Specify the disposition of the MVS dataset

status	NEW	or
	MOD	or
	OLD	or
	SHR	or
	SCR (for scratch)	

Note: the SCR value in BCOM specific. It creates a new dataset if none exists; or delete the existing dataset and create a new one.

Normal-disp	KEEP	or
	DELETE	or
	CATLG	or
	UNCATLG	
Abnormal-disp	KEEP	or
	DELETE	or
	CATLG	or
	UNCATLG	

Any of these three parameters can be overwritten as defaults are defined for each one.

```
LABEL=(NN,TYPE,Password Protection, IN|OUT,
       RETPDnnnn | EXPDT=yyddd or yyyy/dddd)
```

Specify information pertaining to a tape on the MVS. Used when creating tape datasets on MVS; or when referring to a specific volume sequence number of an uncatalogued MVS tape acting as the FROM-FILE.

NN 2 numeric bytes and is the sequence number.

TYPE

SL	OR
NL	OR
BLP	OR
NSL	

Password Protection

NOPWREAD Read only on the DSN
PASSWORD Passowrd need 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 dataset. Valid only when creating the disk dataset on the MVS.

TYPE

CYL	or
TRK	

Note: Allocation of space in a cylinder or trace unit. 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 Reserved space for the member names of partitioned datasets.

RLSE To release all unused space when a dataset is closed, after having been opened for output and written into.

```
UNIT=([xxxx][,number][,DEFER])
```

Specify the UNIT upon which the MVS dataset resides.

When creating a tape on MVS, UNIT should specify the TAPE indetifier.

If you wish to control the placement of the new datasets on MVS, UNIT should be specified.

If MVS datasets are cataloged, UNIT need not be specified.

xxxx	Hardware address Device type Group name	or or
number	The number of volumes to be mount on (1-59) in parallel.	
DEFER	Indicate to delay the allocation of the resource at opening of the file.	

```
VOL=(MVS-dataset-volume-information)
```

Specify the volume upon which MVS dataset resides.

Users may optionally specify the maximum volume count a transfer can use, by entering a 2-digit number instead of the volume list.

The format of this attribute is:

- VOLUME LIST The format of this attribute is

  ```
  VOL=(AAAAAA,BBBBBB,CCCCCC)
  ```

  With upto 3 volumes of up to 6 characters each.

- VOLUME COUNT The format is

  ```
  VOL=mm
  ```

  Where *mm* defaults to 5 if not specified.

```
FROMKEY=string | 'string' | X'hhhhhhhhhhhhhhhhhh'
TOKEY=string   | 'string' | X'hhhhhhhhhhhhhhhhhh'
```

FROMKEY and TOKEY specify the range of primary key values of the records to be selected from a VSM KSDS file.

FROMKEY value is included, TOKEY excluded.

Both attributes are optional.

```
FROMADDRESS=9999   | X'hhhhhhhh'
TOADDRESS=9999999  | X'hhhhhhhh'
```

Attributes are valid ONLY if file organization is VSAM ESDS.

FROMADDRESS and TOADDRESS specify the range of record address for the records to be selected from a VSAM ESDS file.

FROMADDRESS value is included, TOADDRESS excluded.

Both attributes are optional.

```
FROMNUMBER=9999999 | X'hhhhhhhh'
TONUMBER=9999      | X'hhhhhhhh'
```

Valid ONLY if file organization is VSAM KSDS, ESDS, RRDS.

FROMNUMBER and TONUMBER specify the range of record number of the records to be selected from a VSAM RRDS file.

FROMNUMBER value is included, TONUMBER excluded.

Both attributes are optional.

```
SUBSYS=(SUBSYS Name)                              OR
SUBSYS=(SUBSYS Name,<PARM1>,<PARM2>,<PARM3>)
```

Specify 4 bytes SUBSYSTEM; OR

Specify 4 bytes SUBSYSTEM with up to 3 PARMS, each can be 15 bytes max.

```
DATACLAS=(Data Class)
```

Specify SMS Data Class.

```
MGMTCLAS=(Management Class)
```

Specify SMS Management Class.

```
STORCLAS=(Management Class)
```

Specify SMS Storage Class.

```
SELECT=(member-list)
```

Specify the member in the local file that will be transferred by an SP Request Type (for transfers between MVS's only).

`EXCLUDE=(member-list)`

Specify the member in the local file that will not be transferred by an SP Request Type (for transfers between MVS's only).

`COPYPARM=(Replace | Noreplace, List | Nolist)`

Use to overwrite Default option for an SP Request Type (for transfers between MVS's only).

Replace will replace members with the same name in the target PDS.
Noreplace will skip identical members in the target PDS.

Note:1 *SELECT and EXCLUDE are mutually exclusive. Therefore, you can use one or the other in a request, but not both.*

Note 2: *To copy an entire PDS, do not use either SELECT or EXCLUDE.*

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. Begin with a % sign.

`REMOTE-FILE file-name`

Identify the remote file using the file specifications or name syntax of the remote platform.

Note: *If the remote file is on NSK Guardian, syntax is*

```
$Guardian-file-name                OR
DNS-ALIAS=alias-file-name
```

For all other remotes, you must enter the file name.

`REMOTE-ATTRnn identifier=value`

where various sets of identifiers can be used, depending on the platform type:

For an MVS node:

the attribute definitions are the same as depicted for the LOCAL-ATTRnn.

Note: *As for the SELECT= and EXCLUDE= attributes, when are used as remote attributes on an RP request type, they reference member(s) to be selected or excluded from the remote PDS (on MVS only).*

For an NSK Guardian node:

`BRPARAMS = <NSK Guardian Backup/Restore options>)`

For requests type FB/FR only, specify the fileset and option of NSK Guardian Backup/Restore utility. Please refer to NSK Guardain manual for details.

The backup/restores commands can be retrieved from a user edit file. By specifying:

```
BRPARAMS=INFILE <file-name>
```

The edit file must contain the whole backup/restore command, using =BCOM-TAPE as the tape device name.

`JOB-WAITED = YES | NO`

Indicate whether a job submission request should wait for the termination of the job before completing. Default is NO.

`OUTFILE = <NSK Guardian file name>`

Specify the name of the OUTFILE to be used by TACL for Local-Job, File-Backup or File-Restore type requests.

`PARAM = <value>`

Allow you to pass parameters to the I/O process or job being created by BCOM. For example, to pass data management attributes to FUP, SQLCI, like:

SORTED	always included when transferring to NSK Guardian
APPEND	adding to Entry-Sequenced or Relative files
SHARE	only valid for Send-File request type
PROTECTED	the default; as opposed to SHARE
VARIN/VAROUT	for sending data to or receiving from a process.

`PROCESS-PRIORITY = <value>`

To assign a Guardian process priority to the I/O process, backup/restore utility, or NSK Guardian TACL job referred by the request.

`REPORT = ON | OFF`

Specify ON if your interpretation is to have a print control characters' OFF for no print control character interpretation. OFF is the default.

For a Windows NT node:

1. For File Transfers

`TYPE=<Data-Structure>`

This parameter can be valued UNSTRUCTURD (default), TEXT or RECORD.

`RECFORMAT=<Record-Format>`

This parameter can be valued FIXED (default) or VARIABLE.

`OFLAF=<Writing-Method>`

This parameter can be valued SUPERSEDE (create or replace – default), APPEND, CREATE (if does not exist) or TRUNCATE (replace).

`RECLEN=<Record-Length>`

This parameter must be valued between 1 to 32767 (default).

2. For Job Submission

`JOB-WAITED = YES | NO`

Indicate whether a job submission request should wait for the termination of the job before completing. Default is NO.

`PARMS = <Parameters>`

Parameters needed for the job.

IV. Keywords related to OTHER ASPECTS of the request

`ABNORMAL-END #request-name`

#request-name is the name of request to be submitted if the current request terminates unsuccessfully.

`AUTO-RESTART YES | NO`

Instruct BCOM to automatically start a transfer from the last checkpoint value. 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`

record-count is the interval after which a checkpoint will be taken.

`COMPRESSION YES | NO`

where *YES* specifies that compression should be used for this file transfer.

`COMMENTnn comment-line`

To provide nn comment lines within a request definition. A Maximum of 5 contiguous comment-lines are supported.

`FIELD offset,length | ALL`

where the parameters specify locations within a record for which no translation will be performed. Not valid for Backup or Restore file transfers. A warning will be issued at file transfer time if an offset, length is specified which exceeds the record size of the file.

Note: *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.*

`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 9.9. 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 is 2.

The following table indicates when these parameters are mandatory or optional, depending on the type of request they are used with.

REQUEST PARAMETERS	MANDATORY PARAMS for REQUEST TYPE								DEFAULT VALUE (comment)
	FILE TRANSFER		PDS TRANSFER		LOCAL JOB (LJ)		REMOTE JOB (RJ)		
	SF	RF	SP	RP	local data	rmt data	local data	rmt data	
AUTO-RESTART									NO
CHECKPOINT					must be ZERO	inv	inv	inv	NOT USED
COMPRESSION					must be NO	inv	inv	inv	NO
NORMAL-END									
ABNORMAL-END									
COMMENT									
FIELD					inv	inv	inv	inv	ALL
LOCAL-LOCATION	√	√	√	√	√	√	√	√	Note 2
LOCAL-ATTRnn									
PRIORITY-CLASS	√	√	√	√	√	√	√	√	5
ROUTING-CLASS	√	√	√	√	must be '00'	√	√	√	
TYPE	√	√	√	√	√	√	√	√	
LOCAL-FILE	√	√	√	√	√		√ Note3		
REMOTE-LOCATION	√		√	√	same as Local-Lctn	√	√	√	
REMOTE-FILE	√		√	√	inv	√ Note3		√ Note3	
REMOTE-ATTRnn	Note1				inv	Note1		Note1	

inv = parameter is invalid

√ = accepts such transaction

Notes:

1. see remote attributes for remote nodes in previous pages.

2. defaults to the location of the BCOM monitor that is opened by the current BINT process; the file parameter determines where the input JOB data resides.

3. if the remote platform is NSK Guardian, the fileset is specified on the BRPARAM= keyword of an ATTRnn statement; multiple ATTRnn may be used to continue the file set.

Example 1:

The following commands illustrate how a current request definition is modified using SET commands, and then a new request with the updated current definitions is added in the Request File.

```
SHOW

TYPE              = RECEIVE FILE
ROUTING-CLASS     = 03
PRIORITY-CLASS    = 01
LOCAL-LOCATION    = %MVS2
LOCAL-FILE        = USR1.FILE2.TEMP
LOCAL-FILE-ATTRIBUTES:
 01 DISP=NEW
REMOTE-LOCATION   = %TAND2
REMOTE-FILE       = $DATA01.V40.TEMP
REMOTE-FILE-ATTRIBUTES:
AUTO-RESTART      = YES
CHECKPOINT        = YES
COMPRESSION       = YES
ABNORMAL-END      = NONE
NORMAL-END        = NONE
OWNER ID          = USR1

SET  TYPE             SF
SET  LOCAL-LOCATION   %MVS1
SET  REMOTE-LOCATION  %TAND1
SET  ROUTING-CLASS    01
SET  LOCAL-FILE       USR1.FILE1.TEMP
SET  REMOTE-FILE      $DATA05.V40.TEMP
SET  LOCAL-ATTR01     DISP=SHR
SET  AUTO-RESTART     NO
SET  CHECKPOINT       NO
SET  COMPRESSION      NO
SET  NORMAL-END       #RQST2


SHOW

TYPE              = SEND-FILE
ROUTING-CLASS     = 01
PRIORITY-CLASS    = 01
LOCAL-LOCATION    = %MVS1
LOCAL-FILE        = USR1.FILE1.TEMP
LOCAL-FILE-ATTRIBUTES:
 01 DISP=SHR
REMOTE-LOCATION   = %TAND2
REMOTE-FILE       = $DATA05.V40.TEMP
REMOTE-FILE-ATTRIBUTES:
AUTO-RESTART      = NO
CHECKPOINT        = NO
COMPRESSION       = NO
ABNORMAL-END      = NONE
NORMAL-END        = #RQST2
OWNER ID          = USR1

ADD #EXAMP1
REQUEST ADDED
```

Example 2:

The following commands illustrate how the current request definitions can be set using the definitions of an existing request in the Request File, and then modified with a few SET commands. Finally, a new request is added in the Request File using the new current definitions.

```
SET LIKE #EXAMP1
SHOW

TYPE              = SEND-FILE
ROUTING-CLASS     = 01
PRIORITY-CLASS    = 01
LOCAL-LOCATION    = %MVS1
LOCAL-FILE        = USR1.FILE1.TEMP
LOCAL-FILE-ATTRIBUTES:
 01 DISP=SHR
REMOTE-LOCATION   = %TAND2
REMOTE-FILE       = $DATA05.V40.TEMP
REMOTE-FILE-ATTRIBUTES:
AUTO-RESTART      = NO
CHECKPOINT        = NO
COMPRESSION       = NO
ABNORMAL-END      = NONE
NORMAL-END        = #RQST2
OWNER ID          = USR1

SET  LOCAL-FILE   USR1.FILE3.TEMP
SET  REMOTE-FILE  $DATA03.V50.TEMP

SHOW

TYPE              = SEND-FILE
ROUTING-CLASS     = 01
PRIORITY-CLASS    = 01
LOCAL-LOCATION    = %MVS1
LOCAL-FILE        = USR1.FILE3.TEMP
LOCAL-FILE-ATTRIBUTES:
 01 DISP=SHR
REMOTE-LOCATION   = %TAND2
REMOTE-FILE       = $DATA03.V50.TEMP
REMOTE-FILE-ATTRIBUTES:
AUTO-RESTART      = NO
CHECKPOINT        = NO
COMPRESSION       = NO
ABNORMAL-END      = NONE
NORMAL-END        = #RQST2
OWNER ID          = USR1

ADD #EXAMP2
REQUEST ADDED
```

Example 3:

The following is an example of a typical DISP command.

```
SET  REMOTE-ATTR01  DISP=(NEW,CATLG,DELETE)
SET  REMOTE-ATTR02  SPACE=(CYL,(10,2))
SET  REMOTE-ATTR03  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 cataloged at transfer termination. Notice that when going to DASD with a DISP of NEW, both SPACE and DCB are usually required.

Example 4:

This SET LOCAL-ATTR example would be used when selecting the second dataset on a tape volume. This would be required only if the dataset was not cataloged.

```
SET  LOCAL-ATTR01  LABEL=(2,SL)
SET  LOCAL-ATTR02  VOL=D35P12
SET  LOCAL-ATTR03  UNIT=TAPE
```

Example 5:

An example of the DCB command for a file transfer to an MVS generation data group (GDG).

```
SET  REMOTE-ATTR01  DCB=(MODEL.JCL,BLKSIZE=32000)
```

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
TYPE              = SEND-FILE
RQST-COMMENTS:
01 This is a transfer request of a currently
02 existing file
03 comment line
05 last comment line
ROUTING-CLASS     = 01
PRIORITY-CLASS    = 01
LOCAL-LOCATION    = %MVS1
LOCAL-FILE        = USR1.FILE3.TEMP
LOCAL-FILE-ATTRIBUTES:
 01 DISP=SHR
REMOTE-LOCATION   = %TAND2
REMOTE-FILE       = $DATA03.V40.TEMP
REMOTE-FILE-ATTRIBUTES:
AUTO-RESTART      = NO
CHECKPOINT        = NO
COMPRESSION       = NO
ABNORMAL-END      = NONE
NORMAL-END        = #RQST2
OWNER ID          = USR1
```

Note: *If Local-File DNS-Alias = <alias name> is used, the SHOW command will display the request parameters with DNS-ALIAS = alias-name.*

SHUTDOWN

Please use MVS command CANCEL <BCOM-jobname> to stop the BCOM environment.

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 $transfer-process-name | ALL
```

where:

transfer-process-name 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 monitor.

Considerations:

As stated above, all entities referred to in the START command must be defined as a valid entity occurrence in the BCOM Configuration file. Please consult the *BCOM for MVS Installation and Operations Guide* for details on configuring the environment to BCOM on the MVS.

Please note that the START command must be processed through the BINT interface, in any of the modes described in *BINT- BCOM User Interface* Chapter.

Example:

This START command will start all configured but inactive transfer processes for all configured remote locations.

```
START ALL

$MTL01  PROCESS STARTED
$TOR02  PROCESS STARTED
$CHI03  PROCESS 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 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-name | server | %remote-location |
        #request-name [, #request-name2...]]
```

where:

process-name is the name of the transfer process whose status output is desired. Transfer process names begin with the $ character.

server s the keyword to check the server activity.

remote-location is the logical name of a remote location, whose status output is desired. Remote location names begin with the % character.

#request-name is the file transfer request whose status is to be displayed. Request names begin with the # character.

Considerations:

If no parameter is specified, the Status of the entire ETI-NET File Server environment is displayed, including the Local-location.

Example 1: *Status of a remote location*

```
STATUS   %TAND

REMOTE      RTE      RTE        PROCESS          JOBS                          MAX     ENABLE
LOCATON     CLASS    TYPE         ID       SELECT   STATUS     CONNECTED     RQST    COMPRS

%TAND1      01     STANDARD     %BTD1        01     PROCESSING    YES          1       YES
%TAND1      02     STANDARD     %BTD2        02     DRAINING      YES          2       NO
```

Explanation of Output Fields

ROUTING TYPE indicates whether this transfer process is STANDARD or MULTIPLX. MULTIPLX 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 BPLEX multiplexing option.

JOBS SELECTED 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.

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.

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.

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.

ENABLE-COMPRESS is an indicator that controls whether compression will be performed for requests directed at this transfer process routing class. If SET COMPRESSION is not valued YES, compression will not be performed for any transfer request executing under the control of this transfer process.

Example 2: *Status of Process*

The following STATUS command displays the status of the $EXAMP transfer process.

```
STATUS    $EXAMP

REMOTE      RTE      RTE      PROCESS          JOB                              MAX    ENABLE
LOCATION    CLASS    TYPE     ID       SELECT  STATUS      CONNECTED            RQST   COMPRS

%TAND1      01       STANDARD #EXAMP   01      PROCESSING  YES                  1      NO
%TAND1      02       STANDARD #EXAMP   02      DRAINING    YES                  2      NO

LU          SESSION  REQUEST  NB BLOCKS       STATUS      ERROR                RET    FBK
NAME        ID       NAME     TRANSFERRED                 CODE    REASON

#SNAS3.#POL1 00000000 #RQ01A   0000042         ACTIVE      0000    0000         00     00
#SNAS3.#POL2 00000000 #RQ01B   0000012         ACTIVE      0000    0000         00     00
#SNAS3.#POL3 00000000 #RQ01C   0000064         ACTIVE      0000    0000         00     00
```

Explanation of Output Fields

LU NAME is VTAM Logical Unit name.

NB BLOCKS TRANSFERRED is blocks (as opposed to records) transferred, and displays the current transfer count for jobs executing.

STATUS indicates that LU is idle, transferring, not started or in error.

ERROR will display the SNA error code if BCOM receives an error.

HOLD STATE indicates that this request was the object of a QUEUEHOLD command.

RETRY STATE indicates that BCOM has detected a transfer error which is retryable and is commencing to retry this transfer.

ABORTING indicates that this request was the object of an ABORT command.

Example 3:

The following will display the status of the request name #C1U002.

```
STATUS   #C1U002

REQUEST NAME = #C1U002
QUEUE TIME = 1998-06-27 02:10:04
START TIME = 1998-06-27 02:11:19
END TIME = 1998-06-27 02:14:43
RECORD COUNT = 210
BLOCK COUNT = 21
MESSAGE = TRANSFER SUCCESFUL
```

Other possible values in the “MESSAGE =” field are as follows:

```
abnormal termination
executing
selected
file allocation error
waiting
open error
transfer error
deleted by operator
hold state
retry state
aborting
awaiting completion
```

Explanation of Message Field Values

SELECTED indicates that BCOM has selected this request for initiation and is currently waiting on the remote partner's reply to the allocation message.

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.

WAITING is the status when a request has been queued but has not yet been SELECTED for execution.

AWAITING COMPLETION indicates that BCOM is waiting for acknowledgement of its close request from its remote partner.

WARMSTART

Function:

This command causes a transfer process to attempt to re-establish sessions with its remote partner for all LUs configured under this transfer process.

Syntax:

```
WARMSTART $transfer-process-name.
```

Considerations:

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.

- The transfer process must have been previously started 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 of a transfer process that are not ACTIVE.
- If the LINE, PU or LU is 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.
- A BCOM transfer process abends or has been aborted by operator.

Example:

The following command forces the $EXAMP process to reconnect with the remote platform.

```
STATUS
```

REMOTE LOCATION	ROUTING CLASS	ROUTING TYPE	PROCESS ID	JOBS SELECT	JOBS STATUS	JOBS CONNECTED	MAX RQST	ENABLE COMPRS
%MTL01	01	STANDARD	$EXAMP	00	STOPPED	YES	1	NO
%MTL01	01	STANDARD	$OTHER	N/A	DRAINING	YES	1	YES

```
WARMSTART $EXAMP
$EXAMP HAS BEEN WARMSTARTED

STATUS
```

REMOTE LOCATION	ROUTING CLASS	ROUTING TYPE	PROCESS ID	JOBS SELECT	JOBS STATUS	JOBS CONNECTED	MAX RQST	ENABLE COMPRS
%MTL01	01	STANDARD	$EXAMP	00	EXECUTING	YES	1	NO
%MTL01	01	STANDARD	$OTHER	N/A	DRAINING	YES	1	YES

Appendix B-Messages and codes

This section contains the format and description of the messages issued by BCOM on MVS. All messages are logged to the MVS system console.

There are 3 types of messages issued by BCOM on MVS:

1. General BCOM information, warning and error messages.
2. Prompt for an operator command reply.
3. Response to the operator command.

For list of BCOM for MVS messages, please consult *ETI-NET Messages Manual*.

Message Format 1

```
Message-Id
  Message-Constant    04  characters
  Message-Number      03  characters
  Message-Type        01  character
Filler                01  characters
Request Name          09  characters
Request Type          09  characters
LU Name               09  characters
Dataset-Data
  Dataset Name        44  characters
  Dataset Member      09  characters
Print-Data REDEFINES Dataset-Data
  Remote Id           09  characters
  Class               02  characters
  Form Name           09  characters
  Number of copies    03  characters
  Filler              30  characters
Printd-Data REDEFINES Dataset-Data
  DDname              09  characters
  Filler              44  characters
Message Text          25  characters
Variable-Text         10  characters
```

Total length of the message is 124 bytes

MESSAGE-ID

An eight character code identifies the message. The Message-ID is composed of three data items.

The first four characters indicate which program issued the message. This field will contain the constant 'FS62'. This constant will distinguish BCOM messages from other messages on the console.

The next three digits are used to classify and identify the message.

The messages are categorized in the following way:

100 - 199	Main Task, Functional Message
200 - 299	Main Task, Operational Message
300 - 399	Sub Task, Functional Message
400 - 499	Sub Task, Operational Message

The last character represents the severity of the message:

I - Informative
W - Warning
E - Error

REQUEST NAME

Identify the request that was in session at the time of the message.

REQUEST TYPE

May be one of the following:

```
LCL SF
RMT SF
LCL RF
RMT RF
LCL SP
RMT SP
LCL RP
RMT RP
JCL
PRINTD
PRINT
BJES
```

LU NAME

Indicates on which LU the transfer is taking place.

The following information depends on the request type:

If an error occurs for a request type of Upload or Download the DATASET NAME and DATASET MEMBER will be used to identify the file in the transfer.

If an error occurs for a request type of PRINT or BJES the REMOTE ID, CLASS, FORM NAME, NUMBER OF COPIES, will identify the print file.

If an error occurs for a request type of PRINTD the DDNAME will be supplied.

MESSAGE TEXT

Free format text message used to describe the error.

VARIABLE TEXT

Will contain variable data depending on the text: Abend Code, Card Sequence Number, Record Count, User-ID.

Message Format 2

```
Message-Id
  Message-Constant    04  characters
  Message-Number      03  characters
  Message-Type        01  character
Filler                01  character
Message-Text-1        27  characters
Message-Applid        08  characters
Message-Text-2        01  character
```

Total length of the message is 45 bytes.

Fields Description

MESSAGE-ID

An eight character code identifies the message. The Message-ID is composed of three data items.

The first four characters indicate which program issued the message. This field will contain the constant 'FS62'. This constant will distinguish BCOM messages from other messages on the console.

The next three digits are used to classify and identify the message. The messages are categorized in the following way:

900 - 999 operational control messages

The last character indicates the type of message:

A - Eventual action required

MESSAGE–TEXT-1

Is the following command prompt:

```
'ENTER command for BCOM'
```

MESSAGE - APPLID

Is the VTAM application ID of the BCOM program.

MESSAGE–TEXT-2

Is the closing bracket of the application ID.

Usage

BCOM provides the ability for the operator to communicate with the BCOM monitor via MVS MODIFY (F) and STOP (P) commands. The MODIFY and STOP commands replace the WTOR interface that was used by the previous versions of BCOM.

Please refer to *Appendix A* in the *BCOM for MVS - Installation and Operation* Guide for more details.

Message Format 3

```
Message-Id
  Message-Constant    04  characters
  Message-Number      03  characters
  Message-Type        01  character
Filler                01  character
Message-Applid        08  characters
Filler                01  characters
Message-variable      106 characters
Command-Data REDEFINES Message-variable
  Command             09  characters
  Fixed-text          10  characters
  Operator-Id         08  characters
  Command-message     79  characters
Status-Data REDEFINES Message-variable
  Operator-Id         09  characters
  LU-name             09  characters
  Request-name        09  characters
  Request-type        09  characters
  Num-blocks-xferred  10  characters
  Filler              69  characters
```

Total length of the message is 124 bytes.

MESSAGE-ID

An eight character code identifies the message. The Message-ID is composed of three data items.

The first four characters indicate which program issued the message. This field will contain the constant 'FS62'. This constant will distinguish BCOM messages from other messages on the console.

The next three digits are used to classify and identify the message. The messages are categorized in the following way:

900-999 Operational control messages.

The last character represents the severity of the message:

I – Informative
E - Error

COMMAND-DATA

Can contain one of the following:

1. A message indicating an unknown command
2. A message containing a reply to a command
3. A message containing detailed status information for each LU with an active file transfer

COMMAND

A valid BCOM command entered by the operator at the MVS system console. It can be one of the following:

```
ABORT
DRAIN
SHUTDOWN
STATUS
```

FIXED-TEXT

Contain the constant 'ISSUED BY'.

OPERATOR-ID

ID of the operator who entered the command.

COMMAND-MESSAGE

Indicate what action was taken for the command that was entered.

LU-NAME

Is the logical unit on which the transfer is taking place.

REQUEST-NAME

Is the request name of the transfer.

REQUEST-TYPE

Type of transfer taking place. It can be one of the following:

```
LCL SF
RMT SF
LCL RF
RMT RF
LCL SP
RMT SP
LCL RP
RMT RP
JCL
PRINTD
PRINT
BJES
```

NUM-BLOCKS-XFERRED

Is the number of blocks sent across the LU. Note that these are transmission blocks, and do not correspond to the number of blocks in a dataset.

THIS PAGE WAS LEFT BLANK INTENTIONALLY.