
1 Preface

The data communication system DCM (data communication methods) in BS2000 is part of the data communication system. This system incorporates hardware for structuring networks and connecting general-purpose computers, PCs and terminals, as well as software for configuring, managing and controlling data communication via these networks. DCM provides the services of these networks as functions in BS2000 in a form that is independent of the type of network and its configuration. DCAM (data communication access method) with all other access methods and access options is embedded in the DCM data communication method of BS2000. The DCAM dynamic subsystem is the access method for program-to-program and program-terminal communication. DCM is composed of the modules:

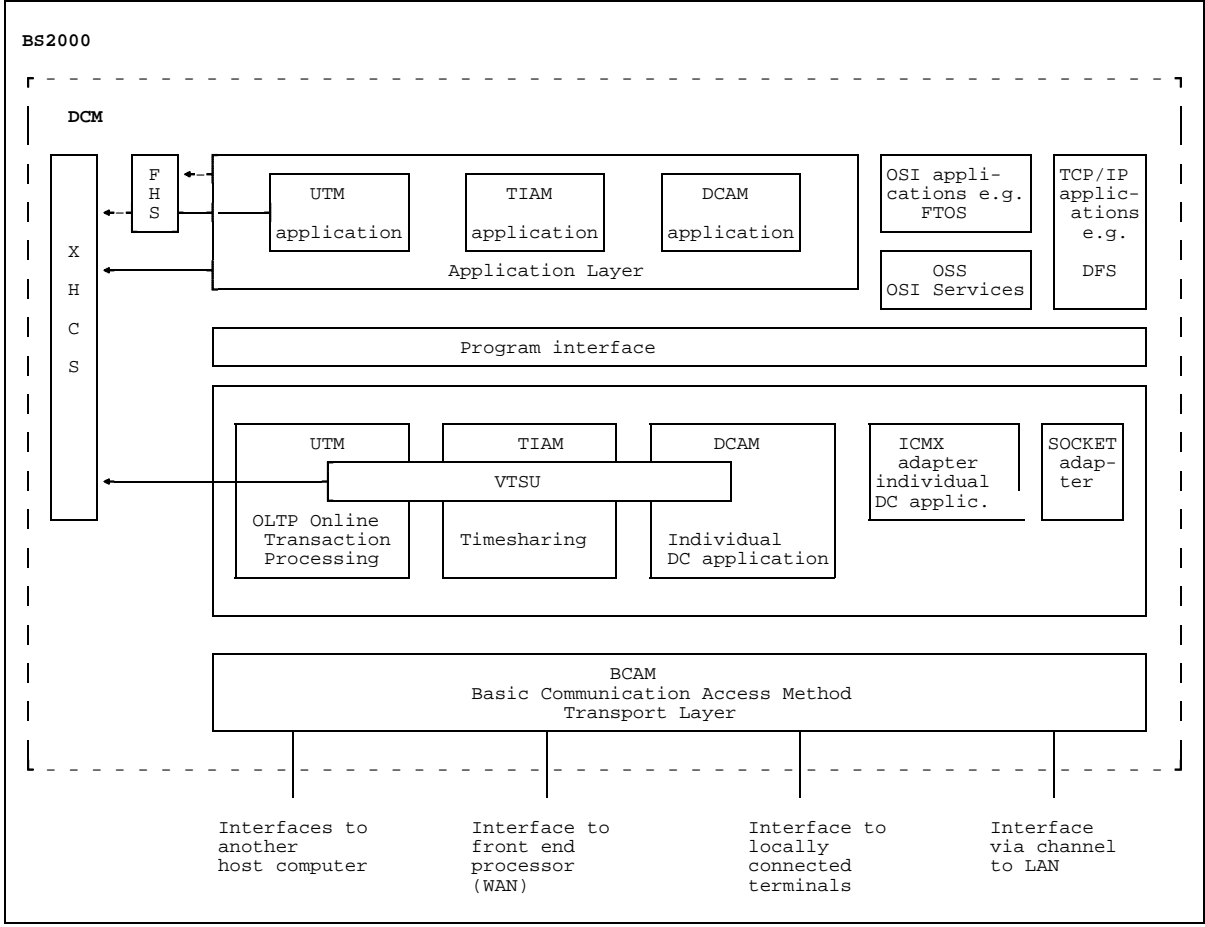
- **DCAM (Data Communication Access Method)** for the implementation of the DCAM interface in ASSEMBLER and COBOL.
- **VTSU (Virtual Terminal Support)** for the implementation of the virtual terminals.
- **TIAM (Terminal Interactive Access Method)** for the implementation of the RTIO ASSEMBLER interface (remote terminal input output).
- **UTM (Universal Transaction Monitor)** for the implementation of the UTM interface.

These modules, in turn, have a defined interface to **BCAM (Basic Communication Access Method)**. The tasks common to all access methods, such as transport management, buffering etc. are implemented in this basic module.

DCM is mounted on decoupled BS2000 interfaces and also provides decoupled interfaces. DCM offers the following facilities:

- **DCAM access method** for communication between programs or programs and terminal.
- support of terminal programming through the application of **virtual terminals**;
- general C program interface **SOCKETS** for accessing TCP/IP networks;
- general C interface **ICMX** offering the OSI transport functionality;
- administration at optional operator consoles, also, if desired, in a DCAM program using the **multiconsole operation module (UCON)**;
- powerful support of the established interfaces for **timesharing (RTIO)**.

DCAM opens up to the user the wide-ranging potential of the data communication system, enabling unrestricted data communication with all the partners (applications and terminals).



Structure of DCM
 XHCS can only be used with BS2000 V10 or higher.

This description of the DCAM (**D**ata **C**ommunication **A**ccess **M**ethod) program interfaces is intended for various user groups:

- O and M specialists and application engineers who want a guide to the scope and capabilities of the interface.
- Programmers (ASSEMBLER, COBOL) wanting to acquire a basic knowledge of the subject in order to understand and use the more detailed information contained in the programming manuals.
- System and network administrators who do not need to be experts on the interface but would like to have a general knowledge of DCAM.

All readers should be familiar with BS2000. The use of DCAM also presupposes a knowledge of either ASSEMBLER or COBOL as well as of the OSI Reference Model.

1.1 Summary of contents

The description of the DCAM communication access method is divided between three manuals:

- "DCAM Program Interfaces"
- "DCAM COBOL Calls"
- "DCAM Macros"

The general description of the DCAM program interface contains basic information for the DCAM programmer, but is also suitable for those wanting a guide to the scope and capabilities of the DCAM interface. The DCAM programmer will then find details on programming in the descriptions of the DCAM COBOL and DCAM ASSEMBLER interface according to the language used.

This DCAM manuals contain the descriptions for both DCAM(ISO) transport service applications and DCAM(NEA) transport service applications. Differences between the two are discussed where applicable. Passages, sections and entire chapters that apply only to DCAM(NEA) transport service applications are indicated by a



at the start of the text.

This manual is subdivided as follows:

- The chapter '**Introduction to the DCAM interface**' contains general information on the DCAM interface, and explains basic concepts and points for consideration in the planning of programs and program systems.
- The chapter '**DCAM functions**' describes the functions of all DCAM calls and notifications.
- The chapter '**Support for virtual terminals**' contains a brief description of format terminals, logical line terminals and edit options.
- The chapter '**DCAM programs**' outlines the coding of DCAM programs in Assembler and COBOL and describes the execution of these programs.
- The **Appendix** lists the DCAM calls and limit values, and shows how a connection is set up by the terminal.

The layout of the function description for all DCAM calls and notifications is identical to the corresponding sections in the manuals for the users of ASSEMBLER and COBOL. This facilitates parallel use of the manuals.

A glossary, list of references and an index are to be found at the end of this manual.

A number of books and guides on computer networks and remote data processing with BS2000 deal with topics related to those discussed in this User Guide. Subjects such as generation and administration, programming communication processors and terminals, and support for virtual terminals are dealt with in separate manuals.

1.2 Changes since the last version of the manual

Support of logical terminals

This chapter entitled 'Support for virtual terminals' is shorter than in previous editions. See the "VTSU User Guide" for a detailed description of the VTSU interface, the VTSU control block, the logical control characters and the status information.

Readme file

Information on any functional changes and additions to the current product version can be found in the product-specific README file. You will find the file on your BS2000 computer under the name `SYSDOC.product.version.READ-ME.D`. The user ID under which the README file is cataloged can be obtained from your system administrator. You can view the README file using the `/SHOW-FILE` command or an editor, and print it out on a standard printer using the following command:

```
PRINT-FILE FILE-NAME=filename,LAYOUT-CONTROL=PARAMETERS(CONTROL-CHARACTERS=EBCDIC)
```

2 Introduction to the DCAM interface

The structure of the DCAM programs is determined by the use of the DCAM interface and access to the communication system. This chapter describes the following:

- the DCAM access method
- traffic relations in the data communication system with DCAM
- explanation of essential basic concepts
- characteristics determining the **performance capability** of the DCAM interface
- the **basic structure** of a DCAM program
- DCAM program **planning**

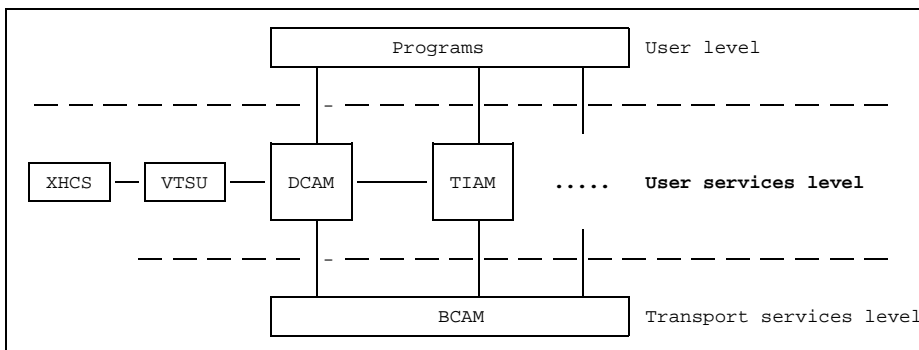
2.1 The Data Communication Access Method DCAM

DCAM offers two different sets of functions:

- DCAM(NEA) transport service functions and
- DCAM(ISO) transport service functions

DCAM(NEA) transport services

In network architecture (NEA), DCAM is assigned to the user services level. This enables programmers and terminal users access to the data communication system.



User services level in BS2000 (NEA)

DCAM(NEA) transport services include the following:

- Provision with partner characteristics
- Definition of rules of communication
- Exchange of user data
- Presentation function for communication with terminals

In order to realize functions for both partners the agencies concerned exchange information adhering to fixed protocols. These protocols provide the framework for DCAM(NEA) transport service applications.

DCAM(ISO) transport services

The DCAM(ISO) transport services enable you to effect data communication on the basis of the transport services standardized by the ISO. The DCAM interface has been adapted in accordance with these transport services.

Overview of the OSI Reference Model

The OSI Reference Model (OSI = "Open Systems Interconnection"), establishes a framework for the classification of services, functions and interfaces. Hence it provides the basis for non-proprietary communication protocols that allow the interconnection of "open systems".

The following diagram summarizes this model with the 7 layers and their respective functions.

For more information please consult the brochure "Ways to Open Communications".

Layer	Designation	Functions
7	Application Layer	Controls the execution of communication functions for an application
6	Presentation Layer	Determines the presentation and meaning of exchanged data
5	Session Layer	Controls the procedure for communication via transport connections
4	Transport Layer	Controls the connections and data transport between the users in the end systems
3	Network Layer	Controls the connections along the transmission path as a whole, i.e. between the end systems
2	Data-Link Layer	Transmission with error recovery along the individual sections of the transmission path
1	Physical Layer	Controls the physical transmission medium

The OSI Reference Model

The layers and their functions are commonly divided into two groups: layers 1 through 4 containing the transport services, and layers 5 through 7 the application services.

DCAM(ISO) provides what is purely a transport service within the framework of architecture for communication on the basis of the ISO standard. This means that DCAM(ISO) does not include some of the functions DCAM(NEA) provides. It should be noted particularly that DCAM(ISO) does not support any protocols on a higher level than the transport service, i.e. no message editing for communication with terminals.

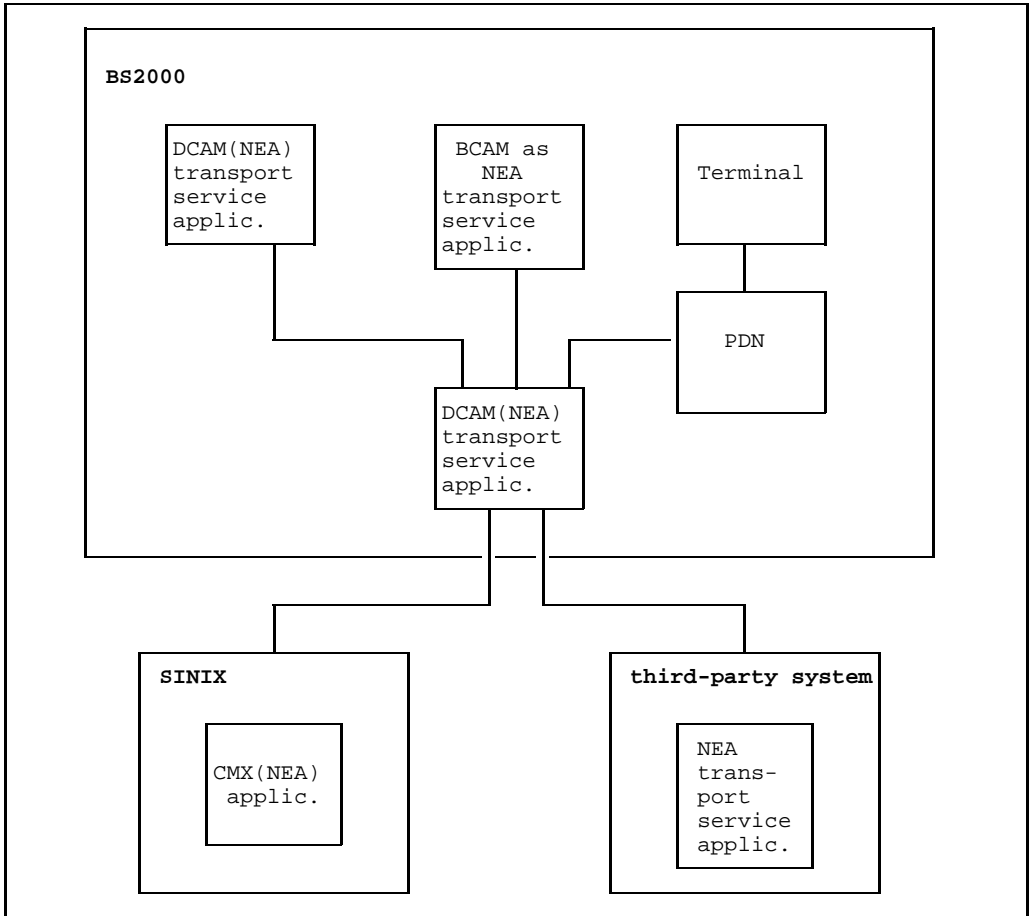
Note that the DCAM(ISO) transport service can be provided on the basis of different communication protocols, e.g. by TCP/IP in conjunction with a convergence protocol.

2.2 Traffic relations in the data communication system with DCAM

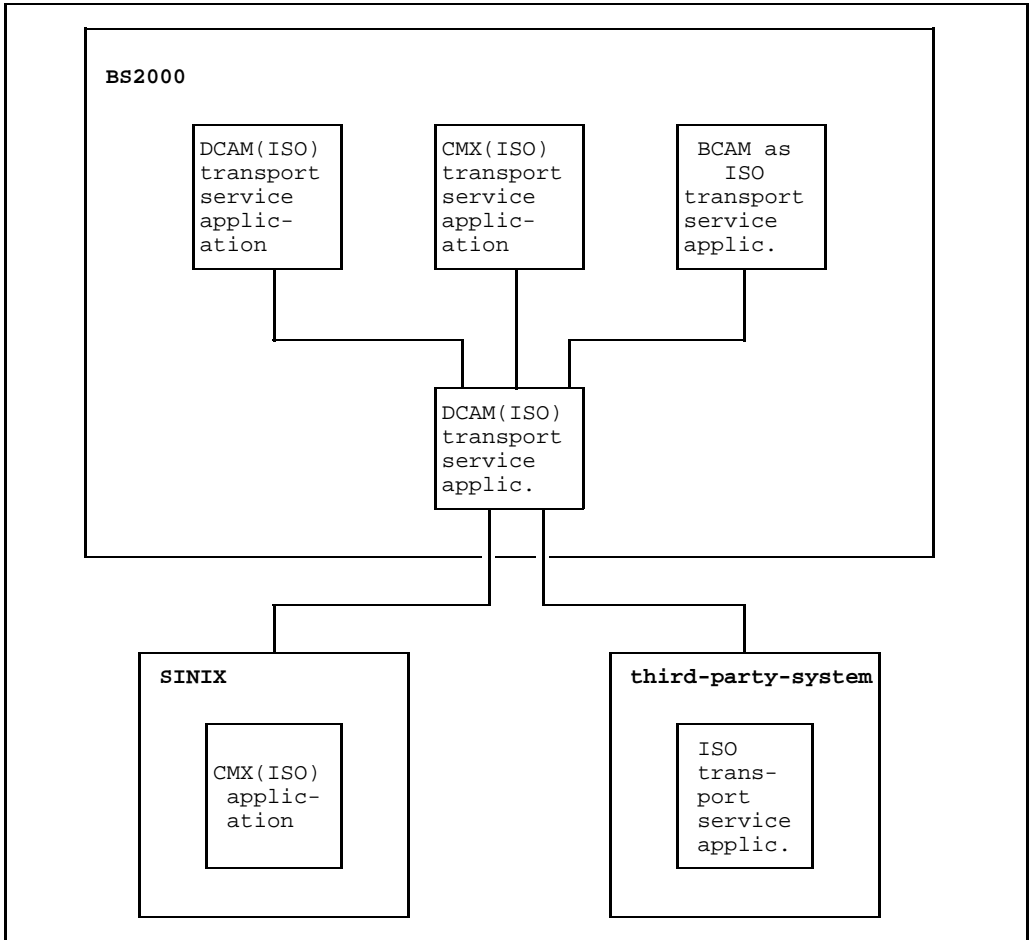
DCAM enables one or more tasks (programs) in the host computer to communicate with any application/program. DCAM(NEA) also permits communication with one or more terminals, and DCAM(ISO) permits communication with CMX applications, if the appropriate /BCMAP commands are implemented. DCAM therefore supports connections to the following:

- other DCAM applications in the same or another BS2000 processor
- CMX applications in SINIX computers
- CMX applications in BS2000 computers
- applications in third-party-computers, if the appropriate transport service is available
- terminals

The links can be established via LANs (local area networks), WANs (wide area networks) or locally.



Interconnections with DCAM(NEA) transport service applications

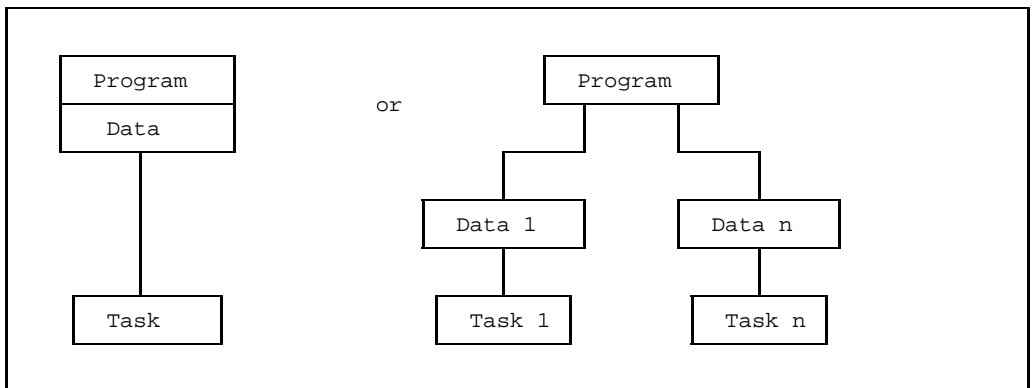


Interconnections with DCAM(ISO) transport service applications

2.3 Basic concepts

2.3.1 Program, data, task

In BS2000 the program and data memory (data memory = data area in the program), and the task controlled by the program, are generally administered as a single unit. For simplicity's sake, therefore, in the following description no distinction is made between these three components. Hence whenever reference is made to 'task' or 'program', this unit is meant. For the transmission and reception of data the task-specific data memory is important, since it is from or in this memory that the transfer is performed. This is why in certain places the data memory is referred to specifically. The specific structural and organizational characteristics of a DCAM program are dealt with in a separate section on page 27ff. Programs controlling several tasks are described on pages 115ff and 125ff, and program execution: DCAM task on page 129ff.



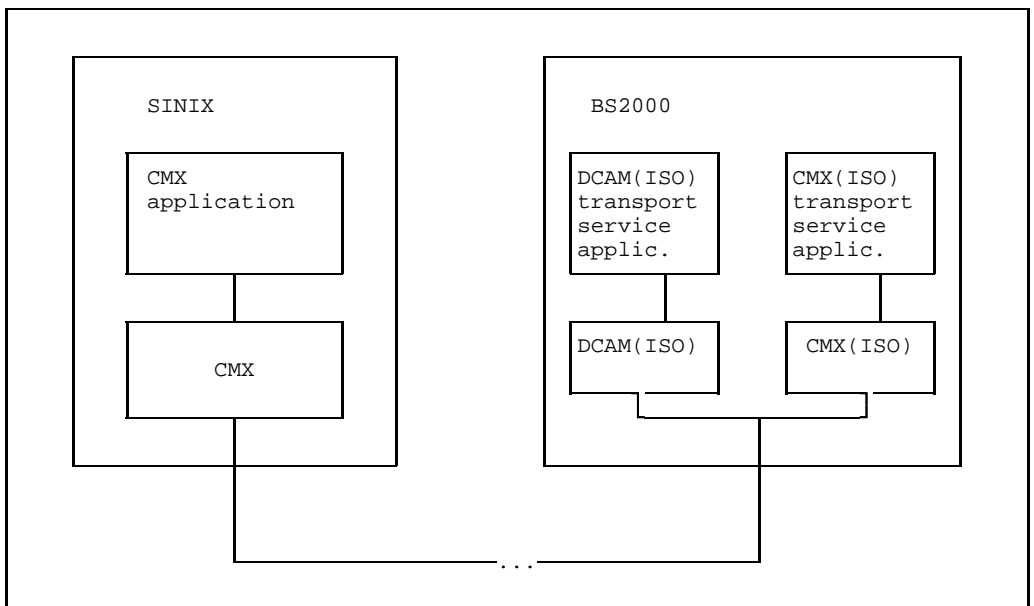
The task as an administrative unit in BS2000

2.3.2 Communication partners

Communication partners in the communication system are applications and, in the case of DCAM(NEA) transport service applications, also terminals.

Applications can be the following:

- DCAM applications
- UTM applications
- PDN applications
- CMX applications
- ISO transport services applications in any computers, e.g. CMX applications in SINIX computers



Communication between a CMX and a DCAM(ISO) transport service application

A **DCAM application** can only exist in a host computer under the control of BS2000. It is defined in at least one DCAM program and generated by the communication access method. It is administered by the communication access method as an addressable unit for incoming and outgoing messages or notifications, and canceled at the request of the program. It is the address at which tasks (programs) or groups of tasks are known to the data communication system. This makes it a communication partner.

The generation of DCAM application type communication partners is a function of the DCAM interface.

A DCAM application can be **opened by one task** if defined as non-shareable or by a **number of tasks** (shareable). The first opening task is the primary task. Secondary tasks may follow if required. More than one DCAM application can be opened in a program. As each one can be considered in isolation, however, this option is not discussed any further in this manual. The name of the DCAM application is laid down when the application is generated (see example in figure below).

This **name** can be specified by the user in YOPEN. If the name is not specified by the user, it is generated by the system, or a predefined name known to the data communication system can be used. The name of the DCAM application must be unique in the host computer in which the DCAM application is generated.

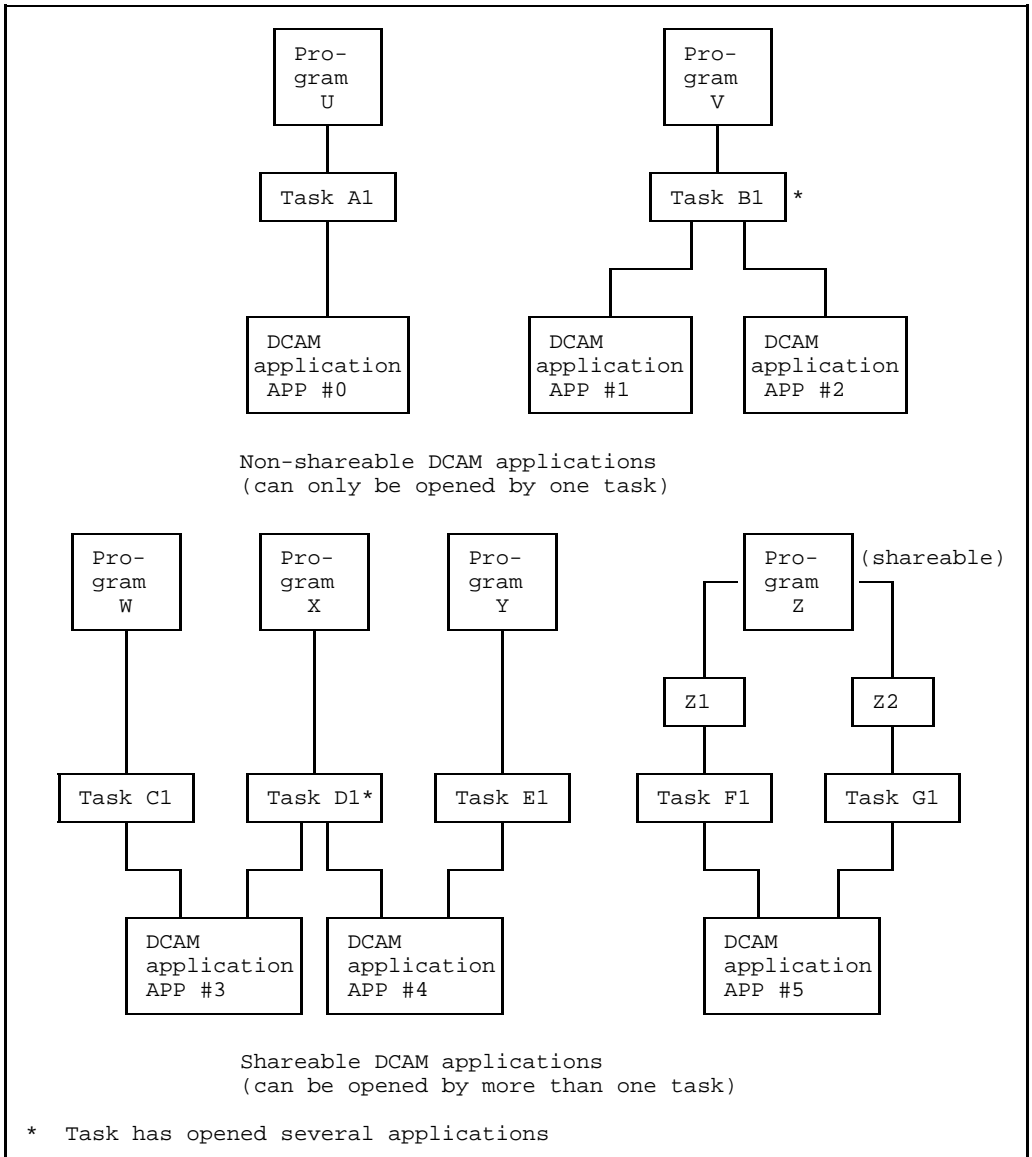


applies only to DCAM(NEA) transport service applications:

The programmer has no influence over the existence of terminal-type communication partners. Such communication partners are generated or removed either at communication system generation time or by means of administration instructions issued by the network administrator.

The **terminal** is characterized by its technical implementation on the one hand and by the person operating it on the other. The interaction of these two elements, i.e. the device and the user of the device, implements the terminal communication partner at a given point in time. This communication partner can only maintain connections to other partners and transmit and receive data via these connections with the means provided by the data communication system.

Those users of the data communication system who can set up connections to other partners are referred to as communication partners in a wider sense. In the more restricted sense, this name is used to denote the two partners linked by an existing connection.



Examples of DCAM applications

2.3.3 Addressing

A partner is addressed via two names: the processor node location of the partner and the name of the communication partner itself.

The name of the processor node and, in the case of DCAM(NEA) transport service applications, the name of the terminal, are defined when the data communication system is generated. The names of the DCAM applications are defined in the program when the applications are opened.

All names used can have a maximum length of 8 characters. The first character must be alphabetic or \$, @, #. Characters 2 through 8 may also comprise the digits 0...9 (ASSEMBLER naming convention). A user application name may only begin with a dollar sign '\$' if the application runs under TSOS.

Note

The name of the communication partner in ISO transport service connections is not validated for conformance to conventions.

In the case of DCAM(ISO) transport service applications and heterogeneous networks it may be necessary to specify different codings or longer partner names. These names are assigned an alias via the /BCMAP console command; this is specified in the program and adheres to the conventions (see the manual "Network Management in BS2000").

2.3.4 Connections

Before communication can take place between two communication partners, a connection has to be set up between them. One partner begins by issuing a **connection request (ACQUIRE)**, whereupon the other may respond with an **acceptance (ACCEPT)**, thereby setting up a connection in the data communication system.

Data can then be sent and received across the connection established. In DCAM(ISO) transport service applications, more than one connection can exist between two applications (parallel connections). Currently, parallel connections can only be established serially.

2.4 Characteristic features of DCAM

2.4.1 Distribution of incoming messages

Messages arriving for a DCAM application can either be distributed via originator-oriented or common receiver queues. These methods are both suitable for shareable and non-shareable DCAM applications.

In the case of DCAM(NEA) transport service applications, distribution can also be via distribution code-oriented queues, but only for shareable DCAM applications.

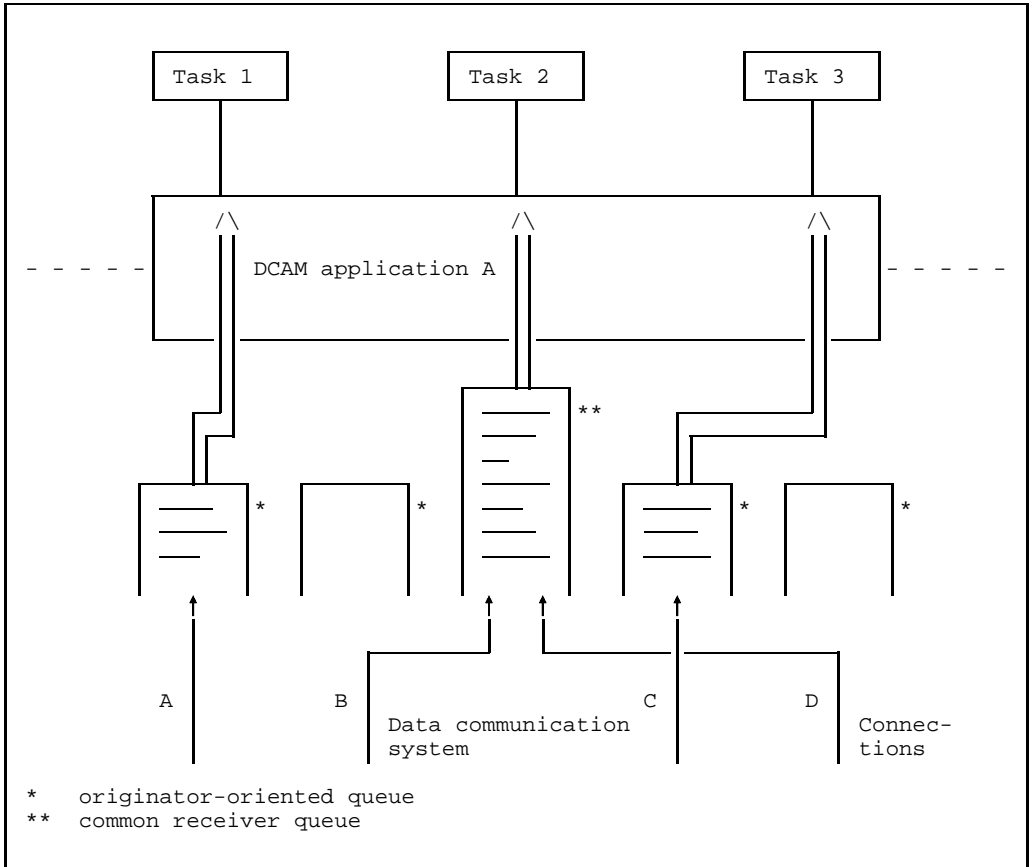
2.4.1.1 Originator-oriented queue and common receiver queue

A queue for incoming messages is set up for each established connection enabling originator-oriented access to the messages. Access in the order of arrival (regardless of the message originator) is implemented by another queue, the common receiver queue. A message is always entered in one of the two queues. The user specifies in the program which queue is to be accessed. This is possible during connection setup and can be re-specified for subsequent processing each time a message is transmitted or received.

Hence, for a definable period of time messages can only be fetched via the originator-oriented queue. In the case of shareable applications this creates the link between a task and a connection. The link is lost once the common receiver queue is used again.

The figure below provides an overview.

Task 1 has specified at connection setup or during the last send or receive operation that it wants to receive messages via connection A, and accesses the associated queue with receive calls. The same applies to task 3 and connection C. Messages from connections B and D are received by task 2 but can also be received by other tasks.



Example of queue access

2.4.1.2 Distribution code-oriented queue



This section applies to DCAM(NEA) transport service applications only.

It is presumed that the task group is controlled by programs with different capabilities, i.e. that each program can only process certain messages. The message/program (task) link is established by means of a code which is contained in the message itself. The user can select the code and specifies specific codes when the connection is established (for a detailed description of the various options refer to page 68ff). A separate queue is created for each code group, and the allocation of the queues to the programs is controlled by the primary task (special macros are provided for this purpose, see page 88ff). A distribution name is defined when the application is opened and this is assigned to the distribution codes in order to link them to a task.

The name can be the same for several tasks, as, for instance, when one program controls more than one task. DCAM then passes on messages to the various tasks in accordance with the FIFO principle (first in - first out: the first entry in the queue will be the first one processed).

When the communication partner enters the code defined at connection setup time or later, he will reach the task whose distribution name has been assigned to this code by the primary task.

Thus the primary task controls

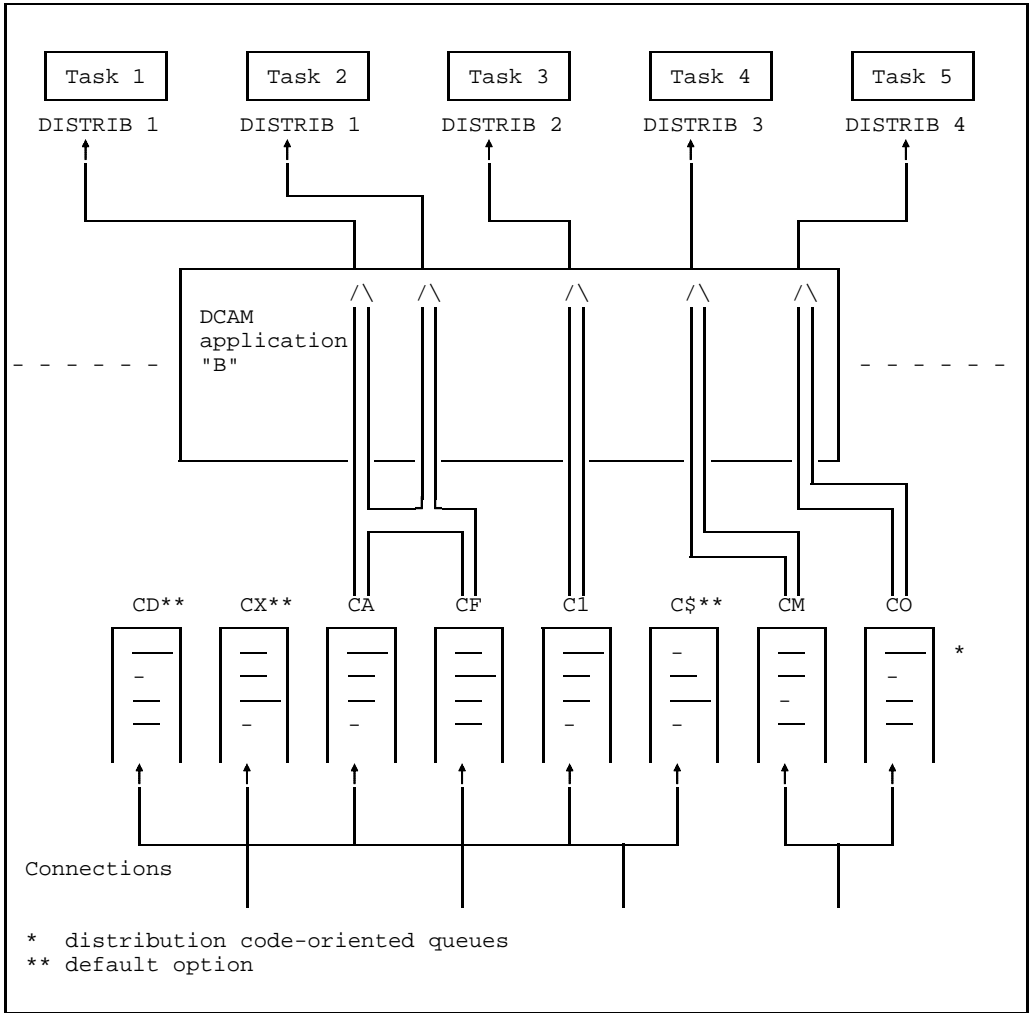
- the assignment of distribution codes to a connection:
when setting up a connection, it determines which codes are to be used and it can redefine these codes for an active connection at any time via a new macro call.
- the assignment of the distribution codes to a task by means of the distribution name.

Messages which cannot be delivered because their code is not allocated or is invalid are delivered to the primary task.

The figure below shows an example of queues for distribution code groups containing only one code each.

As messages with the same code arrive from different partners, originator-oriented access to these messages is also possible.

The distribution name DISTRIB1 is assigned both to task 1 (primary task) and to task 2. At this point the FIFO rule (see above) comes into effect: messages with the codes CA and CF will be passed on in the sequence of their arrival to the next task issuing a receive call. Messages with the codes CD, CX and C\$ are distributed to task 1 because they are not assigned to any other task.



Example of access to distribution code-oriented queues

2.4.1.3 Implicit distribution code



This section applies to DCAM(NEA) transport service applications only.

If a sequence of messages contain the same distribution code it is sufficient to specify this code in the first message. For this purpose the primary task defines a code indicator character which indicates that a **distribution code** is included in a message or message sequence. The distribution code immediately follows this character. In this case it may be no more than 7 characters long.

If messages without a distribution code are received, they are allocated according to the last distribution code received. For subsequent messages DCAM implicitly assumes the distribution code last in force. This applies until the partner sends a message in which a distribution code is explicitly specified.

If a code indicator character has not been defined by the primary task, then a distribution code must be included in each message. Otherwise they will be delivered to the primary task.

The same applies if, in a sequence of messages, at least one message does not contain a distribution code.

2.4.2 Calls and notifications

The functions provided by DCAM consist of calls with which the BS2000 user can effect the execution of certain actions, and of asynchronous notifications (see 111ff) with which DCAM informs the user about certain events in the communication system.

The calls issued to DCAM all begin with the letter Y. They are available either as **macro calls (ASSEMBLER)** or as **COBOL calls**. The calls are terminated after execution or when a defined processing period has elapsed (**synchronous execution**). It is also possible to have control returned immediately after the call was issued (asynchronous execution). Call termination is indicated by an **asynchronous notification** which can be queried in the program. A separate (contingency) routine can also be initiated by the arrival of the notification (see 105ff). This asynchronous processing serves especially to make use of the waiting periods.

2.4.3 Protection against unauthorized access

A DCAM application can be protected against unauthorized **connection of a task** within the host computer. If the application is to be non-shareable, the connection of a secondary task is not possible. In the case of a shareable application, unauthorized connection of a secondary task can be prevented by means of a password.

Predefined applications can already be protected against unauthorized opening (by a user password) in the resource definition file at the system generation stage. This RDF password must be provided by the primary and secondary task.

The unauthorized **setup of a connection** can be prevented within the data communication system. Firstly, by DCAM applications not accepting connection requests, either permanently or for a certain period of time, and secondly, by requiring that a password be specified in a connection request. Finally the user can opt to accept or reject a connection request on the basis of the connection message and/or the address of the requesting partner.

These procedures ensure that inadvertent or deliberate illegal access can be monitored. Passwords can also be dynamically altered for this purpose (see page 92ff).

2.4.4 Express messages



This section applies to DCAM(NEA) transport service applications only.

Another aspect of data security, the solution of conflict cases, is also provided for. Unhindered data transmission is ensured in the data communication system by connection-oriented data flow control and capacity distribution. The existing buffer areas are used by each connection only being allowed to occupy a certain sub-area. Thus, the buffers cannot be completely occupied by the data of one connection. However, to ensure that a connection remains operable in the case of blockage affecting it, short, **fixed-length** express messages can be delivered to the destination as **high-priority messages** which bypass the data flow control. Such messages "overtake" the others, so to speak. If the destination so desires, the express message is transferred to it immediately as an asynchronous notification (cf. page 111ff), otherwise it is entered as far forward in the queue as possible. Express messages cannot be transmitted if the connection is operating with a virtual terminal.

Restriction:

COBOL users cannot receive express messages asynchronous notifications.

2.4.5 Data flow control

The system can reduce the load on an overloaded connection by temporarily interrupting the transmission of messages. This can affect both normal and, in the case of DCAM(NEA) transport service applications, express messages. As soon as the connection is ready for use once more, the user can be informed, by means of a GO signal, that the jam has been cleared and that transmission of messages may be recommenced.

2.5 Program structure

2.5.1 Functions of a DCAM program

DCAM programs serve to implement data communication in BS2000. BS2000. As opposed to timesharing, its object and implementation are completed when the data communication is started. Moreover, DCAM programs usually solve problems which can be solved with specific BS2000 facilities (see table below).

Functions	Supported in BS2000 by:
Dialogs in individual steps or in a sequence of inquiries and responses	<ul style="list-style-type: none"> - Queues for access to specific or any partners - Accompanying information on the connection for transfer with the message (containing, for example, the address of a memory area); cannot be used in COBOL.
Data transfer from programs in the same or in other host computers of the data communication system	<ul style="list-style-type: none"> - Synchronous or asynchronous processing of DCAM calls - Event-controlled processing with EVENTING and CONTINGENCIES
Forwarding of data to exclusive data processing programs in the same host computer and coordination of the cooperation	<ul style="list-style-type: none"> - Use of local or common memory areas with COMMON MEMORY and serialization - Coordination by means of eventing, inter-task communication or user switches
Protection against unauthorized access to programs or program systems or to files	<ul style="list-style-type: none"> - Password protection of the DCAM application - Password protection of the connection - Passwords for the protection of files against unauthorized reading, writing and processing
The following notes apply to DCAM(NEA) transport service applications only.	
Dialog between a few individual terminals and a program	<ul style="list-style-type: none"> - Synchronous processing of DCAM calls - Management of primary tasks (non-shareable DCAM applications)

Functions	Supported in BS2000 by:
Processing of inquiries from a large number of terminals by one program system	<ul style="list-style-type: none"> - Asynchronous processing of DCAM calls with EVENTING and CONTINGENCIES - Management of primary and secondary tasks (shareable DCAM application) - Support of reentrant programming
Formatting of messages for line-by-line output or forms display on CRT screens	<ul style="list-style-type: none"> - Support by means of conversion from virtual terminals to physical characteristics of individual terminal types (VTSU)
Utilization of specific terminal characteristics	<ul style="list-style-type: none"> - Possibility of "physical" programming of terminals
Processing of inquiries in the shortest possible time to provide the terminal user with good response times	<ul style="list-style-type: none"> - Task control in timesharing mode - Rapid file access with user PAM - Reentrant control of programs with shared code in conjunction with good memory usage and low paging rate
Processing of different requests from a terminal in one program system	<ul style="list-style-type: none"> - Message distribution via distribution code-oriented queues

The programmer has at his or her disposal:

the services of the DCAM, TIAM and RBAM interfaces, of the Control System and of the data management system DMS.

It is expedient to subdivide DCAM programs. In the case of shared-code programming, a subdivision into operation code, constants area and dynamic work area is required. This also facilitates documentation and subsequent maintenance.

2.5.2 Basic structure of a DCAM program

A DCAM program includes the following:

- open DCAM application
- establish a connection
- transmit data
- clear connection or close DCAM application

Open DCAM application

To allow a program to be addressed by other communication partners, it must open at least one DCAM application. The name that is defined at that point forms, together with the name of the processor (processor node), the address for this application which must be unique throughout the network. Furthermore, characteristics of the DCAM application are defined, and passwords are specified. The statement that executes this stage is called YOPEN.

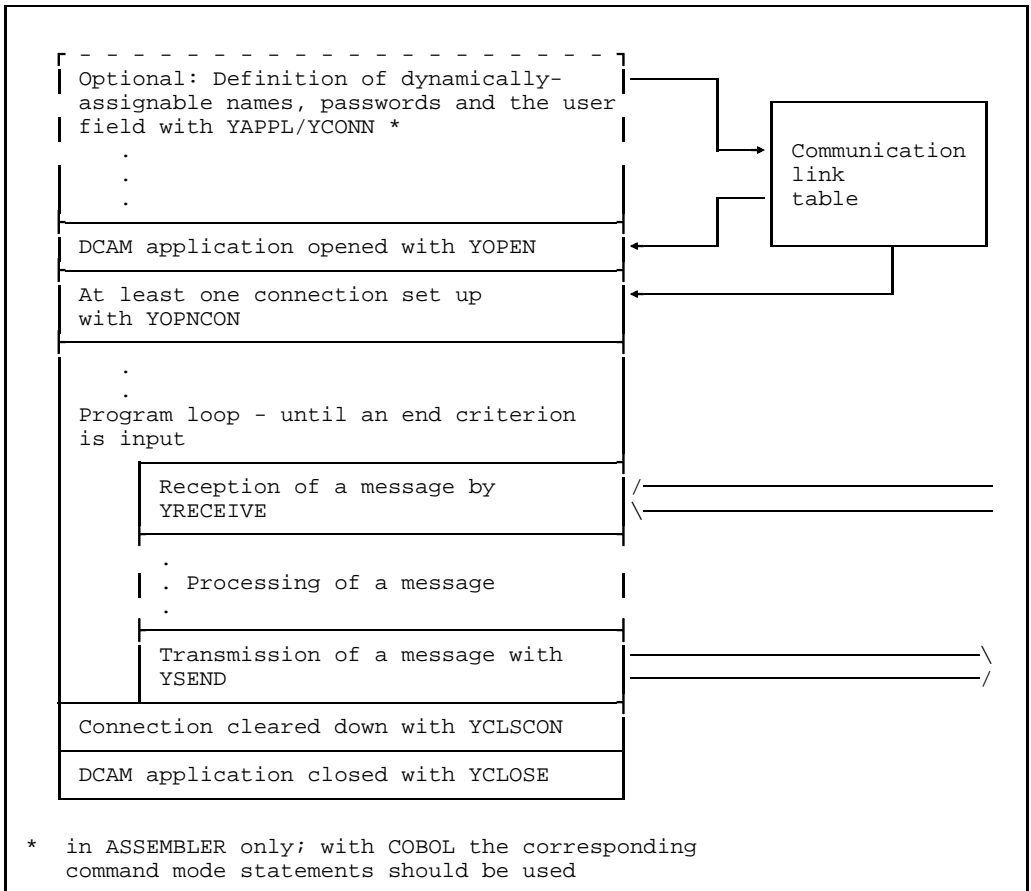
Establish a connection

Before data transmission can take place between two communication partners, one of the partners must send a request to establish a connection to the other, who must accept this request, i.e. a connection must be set up. Buffering and distribution of the messages are carried out according to the specifications made at this point. Moreover, it is necessary to specify what kind of message editing is desired or whether the user will take care of message editing himself. In addition, the communication partners can exchange connection messages. The statement for this stage is YOPNCON ACCEPT (accept connection request) or YOPNCON ACQUIRE (output connection request).

Following the two preparatory stages, the message can be **received** with YRECEIVE and **transmitted** with YSEND.

Once data transmission is completed, the connection is closed down either explicitly by the program with YCLSCON or implicitly by closing the DCAM application by means of the YCLOSE statement, or by the termination of the program. Abnormal termination of a connection e.g. on the request of the communication partners, line failure or failure of a processor, can be reported to the program.

In the case of DCAM(NEA) transport service applications, a connection can also be cleared down indirectly via the terminal by inputting an agreed end criterion which must initiate the execution of YCLSCON in the program.



Schematic layout of a DCAM program

Dynamic name assignment can also influence the structure of the program. The dynamically allocated names, passwords and the user field must be entered in the CLT (communication link table) prior to the opening of the DCAM application or connection setup. A separate program or a leader should therefore contain the YAPPL or YCONN calls or the /SET-DCAM-APPLICATION-LINK and /SET-DCAM-CONNECTION-LINK commands should be used accordingly.

Restriction:

The YAPPL and YCONN calls are not available for COBOL programs.

After each call issued to DCAM, a check must be made to see whether it was properly executed. **Feedback information (FDBK)** is provided for this purpose. The result of the check may be that a call is repeated or has to be executed in a different way, that other measures have to be taken, or that the program run has to be terminated. The error handling routine can immediately follow the call. As the same or similar measures usually have to be taken, it is expedient to generate a central error recovery routine which can be invoked again and again.

2.5.3 Control of primary and secondary tasks

A **primary task**, i.e. the first task to open a DCAM application, contains all the essential DCAM functions. In a task group this task is automatically the **controlling task** which contains the connection function, distribution code control, password definitions and, in the case of DCAM(NEA) transport service applications, the distribution code control, in short everything that concerns the task group as a whole. At YOPEN time, VERIFY can be used to check that the primary task is chronologically the first to open the application. The primary task can transmit and receive messages just like the secondary tasks. When the DCAM application is closed in the primary task, it is also closed for the secondary tasks. The structure of the program controlling the primary task is dependent on various factors that are listed in the table below.

Questions concerning the structure of a program controlling a primary task	Measures taken if the answer to the question is positive
<ul style="list-style-type: none"> - Is the configuration to be operated small? - Is the mean time between the arrival of two messages considerable (low workload)? - Is the job so limited in scope that subdivision into different processing modules is not necessary? 	<p>The primary task should do all the processing; secondary tasks are not required and not permitted (non-shareable DCAM application)</p>
<ul style="list-style-type: none"> - Is the mean time between the arrival of two messages short? - Is the configuration large? - Is a short processing time required? 	<p>The primary task and the secondary tasks should be controlled by a shared-code program. As the primary task must carry out specific functions, this should be provided for in the program. Or alternatively, the program should be structured on a modular basis and only specific modules be utilized and loaded by the tasks.</p>
<p>Applies to DCAM(NEA) transport service applications only:</p>	
<ul style="list-style-type: none"> - Do the inquiries arriving via connections require different types of processing? - Do the distribution codes in the message change? - Are passwords to be used to control the connection of a task to a DCAM application for security reasons? - Is all data to be checked? - Are additional control functions to be implemented? 	<p>The primary task should control distribution code allocation to the secondary tasks and thus message distribution. It will define a password for the connection of secondary tasks. It will check messages that cannot be delivered to a destination within the group. The primary task will have further means at its disposal for the control of the task group within the framework of common memory, eventing, etc.</p>

A **secondary task**, i.e. a task which opens a DCAM application but is not the first to do so, has no influence on the characteristics of the DCAM application, and may have to specify a password in order to join the group. It can use VERIFY to check whether it is opening the application later than the primary task. It can neither establish nor close down connections. It has no control function within the group. Its function is to **transmit** and **receive** messages and to process them. Therefore, the only aspect of significance with regard to the structure of the program is the way in which YSEND, YRECEIVE or YSENDREC calls are issued and processed. Attention must be paid to any controlling actions by the primary task (connection control). If the DCAM application is closed in the secondary task, this has no effect on the other tasks of the group.

2.5.4 Messages and local data units - more-data function

The sum of all the data which a DCAM(ISO) transport service application wishes to send as a logical unit via a connection to a connection partner, is termed **message** (also TSDU = transport service data unit). This message can be of any length.

As a result of fixed requirements in the data communication network (laid down when the data communication network is generated) and the way the transport system is implemented locally (memory management, buffer sizes, etc.) only a limited amount of data can be transferred by the application to the transport system at a time using a local interface call (YSEND, YRECEIVE). This data buffer passed by an interface call is known as a **data unit** (also TIDU = transport interface data unit).

DCAM offers you the option of passing large messages for DCAM(ISO) transport service applications as logical units in a number of interface calls, i.e. in more than one data unit. This facility is called the **more-data function**.

DCAM(ISO) transport service applications can specify when the connection is set up whether or not the more-data function is to be used.

Note

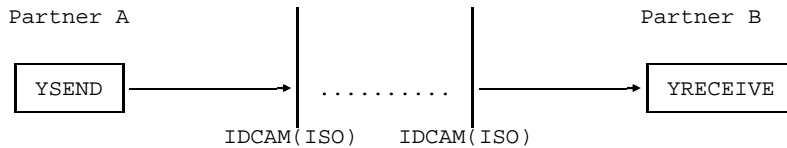
- The more-data function is only set in the DCAM of the user's own computer. It is neither passed to nor negotiated with the connection partner.
- The more-data function determines what data units are passed at the local DCAM interface. It has no bearing on how the "physical" data blocks are split up along the transmission path to the remote transport system.
- The use of the more-data function in one application, does not have any bearing on the data units in which (logical) messages are sent or received by the connection partner.

A few examples have been chosen to illustrate what effect the more-data function can have.

We shall assume that partner A and partner B are both DCAM(ISO) transport service applications. In each example only one (logical) message, sent by A to B, is considered.

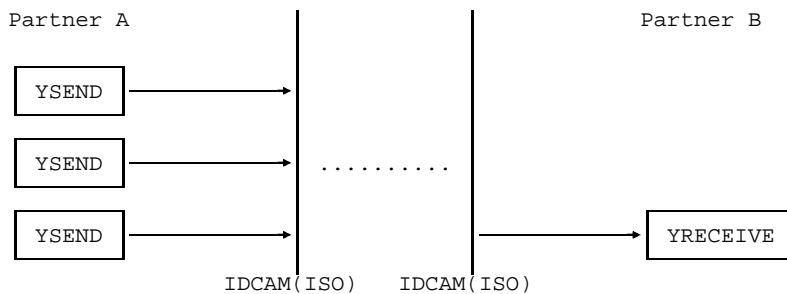
Example 1

- Partner A without the more-data function
- Partner B without the more-data function



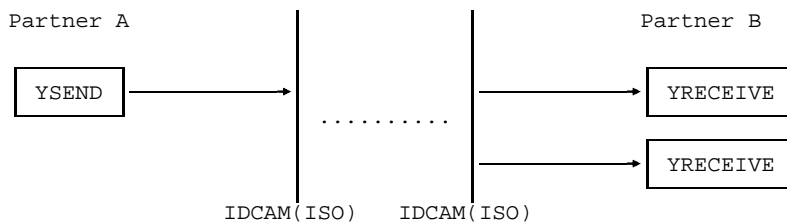
Example 2

- Partner A with the more-data function
- Partner B without the more-data function



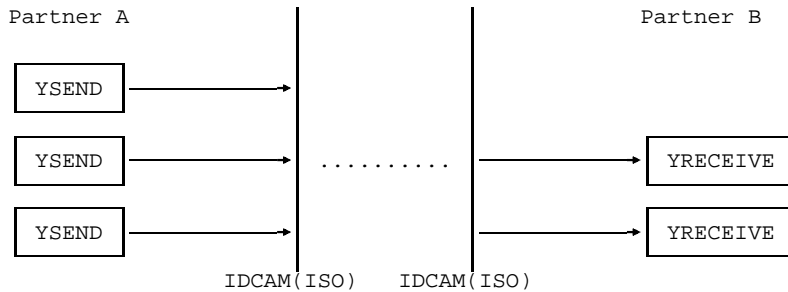
Example 3

- Partner A without the more-data function
- Partner B with the more-data function

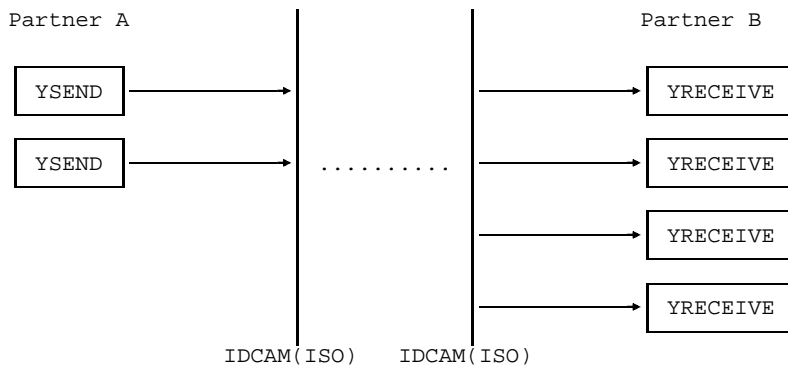


Example 4

- Partner A with the more-data function
- Partner B with the more-data function

**Example 5**

- Partner A with the more-data function
- Partner B with the more-data function



Examples 4 and 5 clearly show that how the data units are divided up at the receive end and the send end need not necessarily be identical.

Note

In the DCAM manuals an explicit distinction between "message" and "data unit" is only made where an understanding of the subject matter requires it.

Wherever the distinction is not important or it is clear from the context what is meant, the expression "message/data unit" is shortened to just "message".

2.5.5 Access to terminals



This section applies to DCAM(NEA) transport service applications only.

An important question in the planning of DCAM programs is how the terminals are to be accessed.

The DCAM programmer has a choice of two methods:

- physical programming
- use of virtual terminals.

Physical programming requires the operation of a terminal with the control characters it understands (see User Guides for individual terminals). The programmer wins considerable flexibility, but has also the inconvenience of operating with control characters. The virtual terminals relieve him of this inconvenience because all essential functions can be implemented with these standardized terminals (but not with locally connected terminals).

The **virtual terminals** are user service software modules in the data communication system. At the logon stage, the communication partners agree on which virtual terminal should be used. Certain aspects have to be clarified in order to do this, such as whether a hardcopy unit is connected to a display terminal or a card reader is to be used.

The programmer has a choice of two types of virtual terminal: the line terminal and the form terminal. For details see 'Logical terminal support' page 95ff and the 'VTSU User Guide'.

2.6 Implementation of distributed processing

The Communication System allows communication between

- applications and applications
- and between terminals and applications (in the case of DCAM(NEA) transport service applications)

Both the terminals and applications can be in a LAN or WAN.

Apart from the connection to terminals, this opens up the following possibilities as partners for a DCAM application:

- a DCAM application
- a UTM application
- a CMX application
- a system application (e.g. \$DIALOG, \$CONSOLE)

2.6.1 A DCAM application as a partner

The communication partner DCAM application is described under 'DCAM functions' as from page 41. This section describes the existence function of a DCAM application (see page 41ff), connection setup between DCAM applications (see page 50ff) and data transmission between DCAM applications (see page 77ff).

2.6.2 UTM application as a partner

Communication between a DCAM application and a UTM application is possible if the DCAM has been generated as a communication partner for the UTM application and if a connection has been established between the two applications. This is true whether the applications are in the same or in different host computers.

There are three different ways of establishing a connection between a DCAM and a UTM application:

- The UTM application has been generated so that when it is started a connection is established. The connection is set up if the DCAM application exists at this moment and explicitly accepts the connection request.
- The UTM application has been generated so that connection requests from the DCAM application are accepted. The connection is established if both applications exist and if the DCAM application has issued an explicit connection request.
- A UTM administration command can be used to cause a connection request to be issued to a DCAM application.

When transmitting data care must be taken that the UTM application has been provided with a structure. The UTM application is composed of subroutines, which can be addressed via the transaction codes which they have been allocated. When a transaction is started the data must be included at the beginning of the transaction code.

3 DCAM functions

The functions provided by DCAM can be subdivided into 4 groups:

- Existence function
- Connection function
- Data transmission function
- Name assignment function

These functions are described below; the layout corresponds to that of the section "Using the DCAM functions" in the ASSEMBLER programmer's manual (see "DCAM Macros") and the COBOL programmer's manual (see "DCAM COBOL Calls"), where the corresponding calls are listed along with the operands required.

3.1 Existence function

The existence function of the DCAM interface performs the following operations:

- **Open or generate a DCAM application (YOPEN).**
This is the basic function of DCAM, and is executed in different ways depending on the type of DCAM application involved.
- **Test the status (YINQUIRE) of a DCAM application.**
This function serves to check the status of a DCAM application.
- **Close a DCAM application (YCLOSE).**
The DCAM application is closed automatically at the end of the program, but this function can close it at any point in time during the program run.



Additional function for DCAM(NEA) transport service applications:

Change the status (YSETLOG) of a DCAM application.

This function can be used to dynamically change the state of the DCAM application, i.e. its readiness to process connection requests.

3.1.1 Open a DCAM application

A DCAM application is generated when the first program opens it. At the same time the application is defined as shareable or non-shareable. A non-shareable application can only be opened by one program and is also closed by that program. A shareable application can also be opened by other programs (secondary tasks). These, however, only link up with the application and have no influence on the definitions made by the first program to open/generate the application (primary task). In the case of DCAM(NEA) transport service applications it is also necessary to specify whether the distribution code-oriented queues are to be used for message distribution. This results in different variants of the YOPEN call.

The ISO attribute must be set in the application control block for all DCAM(ISO) transport service applications. This attribute is then valid for all connections maintained by this application. Connections using NEA services are not possible for DCAM(ISO) transport service applications.

3.1.1.1 Non-shareable DCAM application

A non-shareable DCAM application (NSHARE) can only be opened by one task. A second attempt to open this application will be rejected. Within the bounds of this restriction, it is expedient to lay down further definitions, which may vary depending on a number of requirements which are summarized in the table below.

Requirement	Definition
The name of the DCAM application is not to be generated by the communication access method.	Name of the DCAM application; in the case of COBOL the name of the DCAM application must always be specified by the user
The application is protected by a password in the RDF (APPPW=).	Password USEPW
Asynchronous notifications are to be accepted.	Reference to the identifier of a contingency routine generated when an ENACO call was invoked. If the notification is issued, this routine is activated (not necessary for COBOL).
Specific functions available as from DCAM Version 8.0 are to be used.	Specification of the DCAM version number (DCAMVER).
The following applies to DCAM(NEA) transport service applications only:	
Only the DCAM application is to issue connection requests (NLOGON). In this case the partners must either be terminals or DCAM applications accepting connection requests.	Attribute NLOGON: Connection requests will not be accepted.
It is not known in the program whether and when a partner wishes to log on, or the partner is not known.	Attribute LOGON: Connection requests will be accepted.
Unauthorized connection requests are to be rejected automatically.	Password (LOGPASS), which must be specified by the communication partners.

3.1.1.2 Primary opening of a shareable DCAM application

A shareable DCAM application can be opened by more than one task. The first task to open the DCAM application is the primary task. Message distribution is effected via the originator-oriented or or common receiver queue(s). The name of the DCAM application must be defined in the program. Further definitions are summarized in the table below.

Requirement	Definition
The task is to be the first opening task as it is to fulfill the functions of a primary task. If it is not the primary task, the YOPEN should be rejected.	Check with VERIFY = PRIMARY.
It can be assumed that the task is the first opening task, or all programs linking to the application have the same program logic.	No check, i.e. VERIFY = NO.
The possibility of unauthorized connection of secondary tasks cannot be excluded and is therefore to be prevented.	Password USEPASS, which must be specified by the secondary tasks for subsequent opening. <i>Note:</i> If the DCAM application is already protected by a password (APPW=) in the RDF, the USEPASS password must be identical to that APPW= password. The USEPW operand is then also analyzed in the primary task; it must contain the RDF password.
Asynchronous notifications are to be accepted.	Reference to the identifier of a contingency routine generated when an ENACO call was issued. When the notification arrives, this routine is activated (not required for COBOL).

Requirement	Definition
The following applies to DCAM(NEA) transport service applications only:	
You need to decide whether the DCAM application is to issue connection requests (NLOGON). In this case the partners must either be terminals or DCAM applications accepting connection requests.	Attribute NLOGON: Connection requests will not be accepted.
It is not known in the program whether and when a partner wishes to log on, or the partner is not known.	Attribute LOGON: Connection requests will be accepted.
Unauthorized connection requests are to be rejected automatically.	Password (LOGPASS), which must be specified by the communication partners. <i>Note</i> LOGPASS and USEPASS cannot be changed in an existing DCAM application.
Transport acknowledgments for this application are as a rule processed in the primary task.	PRIMTASK attribute for the transfer of transport acknowledgments.
Transport acknowledgments are processed by the task that sent the message with the corresponding sequence number (SEQNO) and requested an acknowledgment (OPTCD=TACK).	REQTASK attribute for the transfer of transport acknowledgments.
This application does not process any transport acknowledgments because user security procedures are to be implemented.	NOTACK attribute: positive or negative transport acknowledgments are not transferred.
New functions belonging to a DCAM version from 8.0 onwards are to be used.	Specification of the DCAM version number (DCAMVER).

3.1.1.3 Primary opening - use of distribution codes



This section applies to DCAM(NEA) transport service applications only.

The primary task that opens a DCAM application can specify that a distribution code is to be used (SHARE,DISCO) in place of the default option for message distribution. This is especially useful if the programs involved perform different jobs but are accessed by the same partners. All the tasks involved must then define distribution code names permitting association of distribution code(s) and tasks (DISNAME). For further definitions, see table above.

3.1.1.4 Secondary opening

A task which is not the first to open a DCAM application must comply with the definitions laid down in the primary task. It must use the defined name of the DCAM application and also set ATTR=SHARE (shareable DCAM application). The default option provides for message distribution via the originator-oriented or common receiver queues. Further definitions are listed in the table below.

Requirement	Definition
The primary task has defined a password to guard against unauthorized access or the application is protected by an RDF password.	Password (USEPW) for linking up with the DCAM application as specified in the primary task or RDF.
The task must always be a secondary task as this is stipulated in the program logic; if necessary, YOPEN is to be rejected.	Check with VERIFY=SECONDARY.

Requirement	Definition
It can be taken for granted that the task is not the first to open the application, or, alternatively, the program logic checks this requirement.	No check, thus VERIFY=NO.
Asynchronous notifications are to be accepted. <i>Restriction:</i> There are no asynchronous notifications intended for the secondary task at the COBOL interface.	Reference to the identifier of a contingency routine generated when an ENACO call was issued. When the notification arrives, this routine is activated.
The following applies to DCAM(NEA) transport service applications only:	
The primary task specified a DCAM version number on opening the application.	Specification of the same DCAM version number as in the primary task (DCAMVER)

3.1.1.5 Secondary opening - use of distribution codes

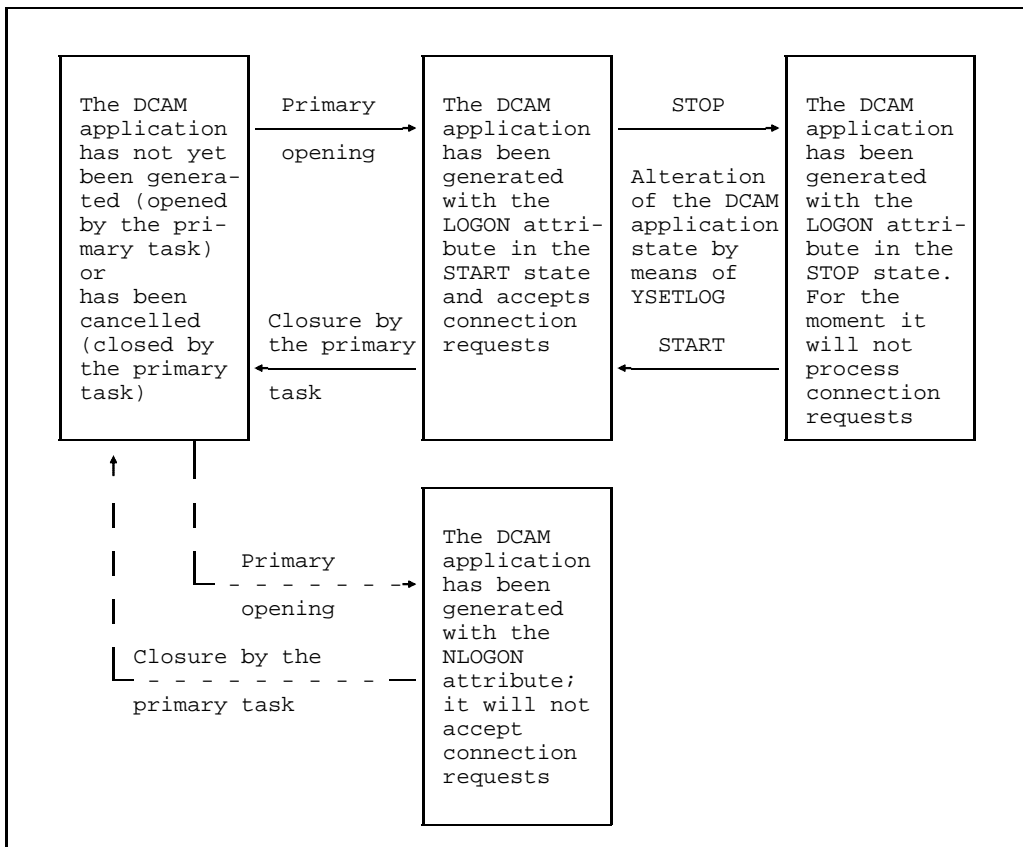


This section applies to DCAM(NEA) transport service applications only.

During secondary opening of a DCAM application, the same message distribution method must be used as during primary opening by the primary task, i.e. the distribution method is uniform within a task group. Distribution codes are used in this variant: the SHARE and DISCO attributes are specified in the primary task; the same DCAM application name and the SHARE attribute must be specified in the secondary task. Furthermore, a name must be defined (DISNAME) to be used for distribution code allocation. All further definitions are listed in the table above.

3.1.2 Altering the state of a DCAM application

 This section applies to DCAM(NEA) transport service applications only.



DCAM application states

DCAM applications processing connection requests (LOGON attribute) can be in two states: START or STOP. The START state causes the requests to be passed on to the DCAM application or, if several request are made, to be entered in a queue. The STOP state means that the primary task has used the YSETLOG call to indicate that it is no longer processing connection requests. The reason may be that the maximum number of connections permitted has been reached or that the primary task will issue requests itself for a while. However, the primary task can restore the START state at any time, i.e. the connection requests are not rejected.

DCAM applications that do not accept connection requests (NLOGON attribute) cannot be altered.

3.1.3 Querying the status of a DCAM application

The YINQUIRE call in the 'APPSTAT' function can be used to query the status of the DCAM application that the user himself has opened, and also the state of a DCAM application which was opened in the same host computer and whose name he knows.

3.1.4 Closing a DCAM application

The DCAM application can be closed in two ways:

- implicitly by terminating the program in which the DCAM application was opened (normal or abnormal termination) or task abortion
- explicitly by means of the YCLOSE call or by means of the /SHUTDOWN, /BCEND or /BCAPPL operator commands.

Explicit closing with YCLOSE will be necessary when, for example, a program is to process several DCAM applications consecutively or if definitions are to be changed during execution. The application can be opened again by the primary task with new definitions (name, attributes etc.) after being closed.

The **secondary task** can close the application at any time without any consequences for the other tasks in the group.

If the **primary task** closes the application,

- the DCAM application is canceled;
- secondary tasks are informed through initiation of the COMEND contingency routine (see page 111ff) or by feedback information for an incomplete call or the next call;
- all existing connections are cleared down.

This also means that data which has already arrived but has not yet been accepted is no longer accessible. Pending connection requests are deleted.

3.2 Connection function

The basic requirement for data transmission is the establishment of a connection between the communication partners after a DCAM application has been opened.

The connection function performs the following:

- set up connection (YOPNCON).
- query entries on partners and connections (YINQUIRE).
- request rejection of connection request (YREJLOG)
- cancel request (YCLSCON)
- close connection (YCLSCON)
- change characteristics of a connection (YCHANGE)

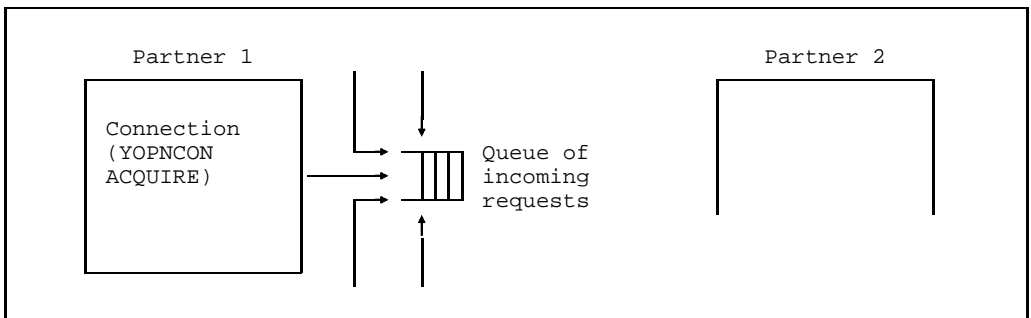
3.2.1 Connection setup: YOPNCON

Two steps are required to establish a connection:

- One partner must send a connection request to another.
- The other partner must accept the request.

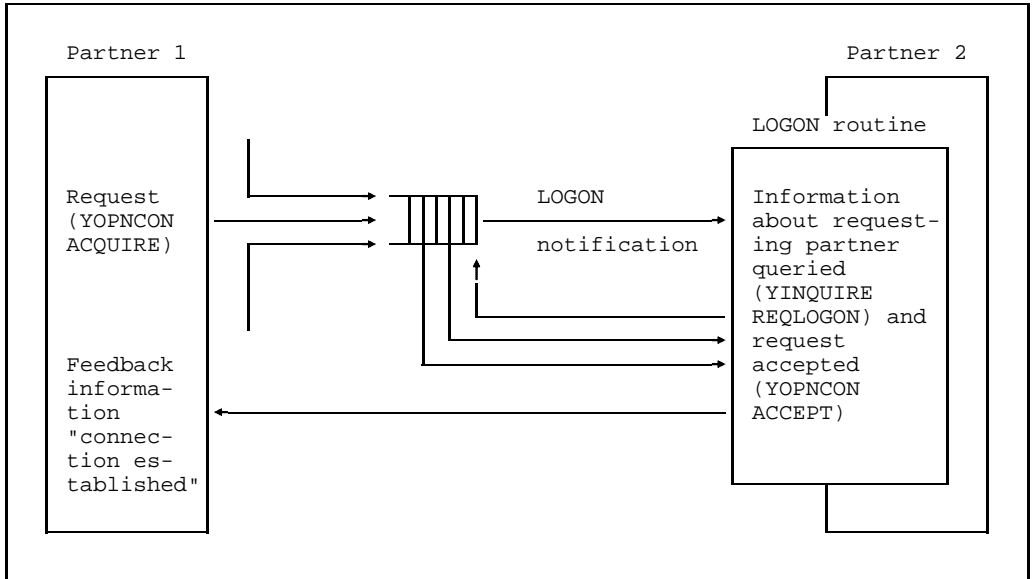
The coordination of these steps is performed by the system:

- The **requests** are forwarded to the partner and, if others are pending, entered in a queue.



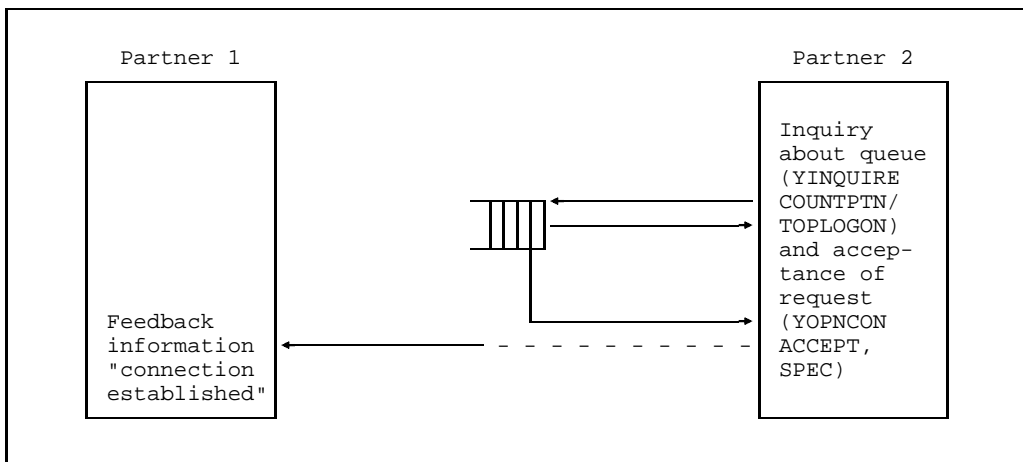
Connection request

- It is possible to have the arrival of a request indicated by an **asynchronous notification** (LOGON notification see page 111ff).
- After the notification has been issued, a **contingency routine** can establish which partner made the request. Following this, the request can be accepted or rejected.



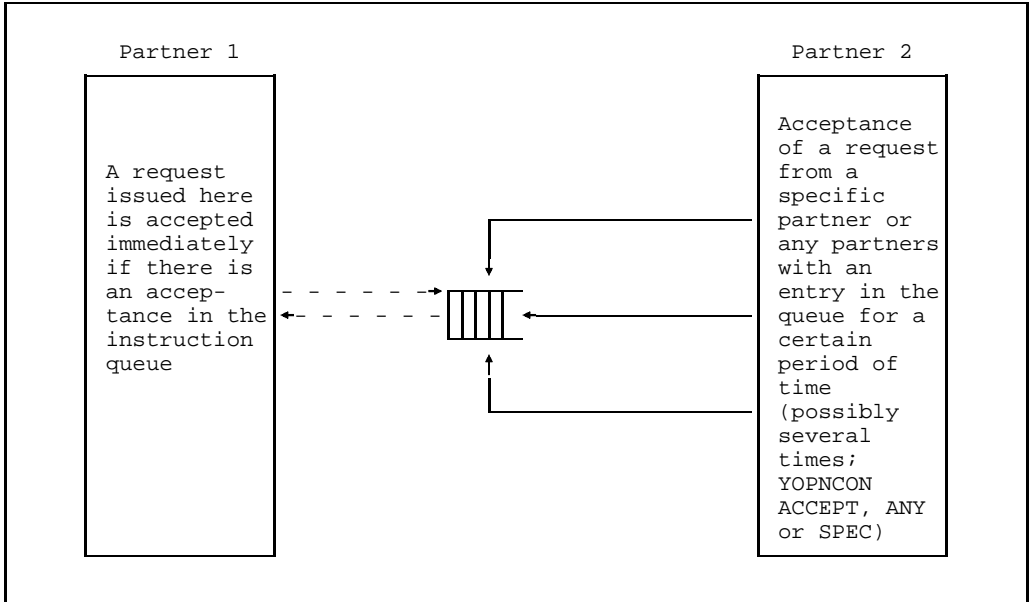
LOGON notification; query and acceptance of request

- The partner can **query** this queue **without** having received a **notification** (see page 66) to discover if entries are present and can, depending on the result, issue accept calls for this request (YOPNCON ACCEPT SPEC).
- He can also reject the request or, by not accepting it, cause the request to be deleted after a period of time defined in the system (/BCTIMES command, see "Network Management in BS2000").



Partner information query and request acceptance

- He can also issue **tentative** accept calls **without a query**. If the request arrives within the specified period of time, it is accepted. A number of this type of acceptances are entered in a queue if the requests have not yet arrived. They may be entered for a specific (SPEC) or an arbitrary (ANY) partner.

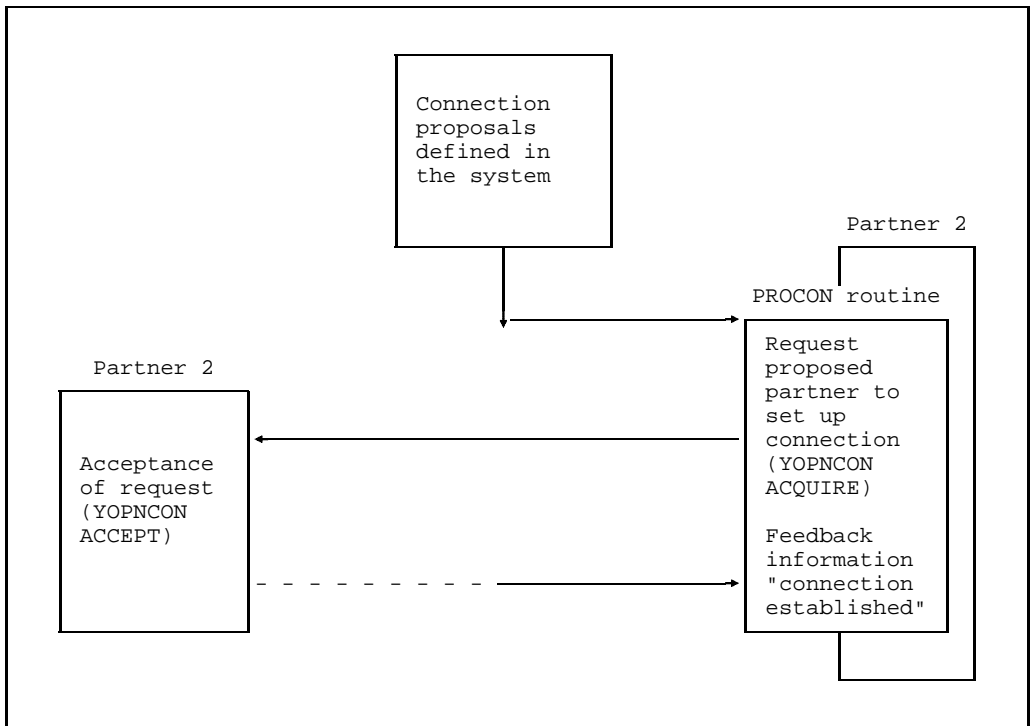


Acceptance of a request

- A further possibility is to assign partner names to a DCAM application **when the communication access method is generated and started** (see XSTAT macro in the manual "Generating a Data Communication System"). When the DCAM application is opened, or a /BCIN command referring to the processor node of the proposed partner is issued by the administrator, the named partners are proposed for connection setup by means of the PROCON notification. As a result of the notification, a request must be issued (YOPNCON ACQUIRE). The connection is only established if the proposed partner accepts this request.

Restriction:

PROCON notifications are not issued to COBOL programs.



PROCON notification, request to proposed partner and acceptance

- Parallel connections can only be set up "serially", i.e. a new parallel connection can only be established once the previous connection setup has been completed (DCAM(ISO) transport services only).



The rest of this section applies to DCAM(NEA) transport service applications only.

- When generating and starting the communication access method, **connections** (XKON macro), as well as connection proposals, may be predefined.

These connections are already established whenever the communication access method is started. However, data transmission is not possible until the primary task issues a call which is usually used as a connection request (YOPNCON ACQUIRE).

If the partner is a terminal, it is sufficient for this to be activated and operational.

Every communication partner that issues a logon call has to define the desired connection characteristics. When a connection request is made, the partner is informed of that part of the connection characteristics which affects him. This part consists of:

- Type of message editing used (EDIT)
- Specification of the partner initiating data communication (PROC)

The figure below shows how this information is requested and defined:

Partner 1 sends partner 2 a connection request (YOPNCON ACQUIRE), and also defines proposals for EDIT and PROC to partner 2. Partner 2 can react to the call from partner 1 in two ways:

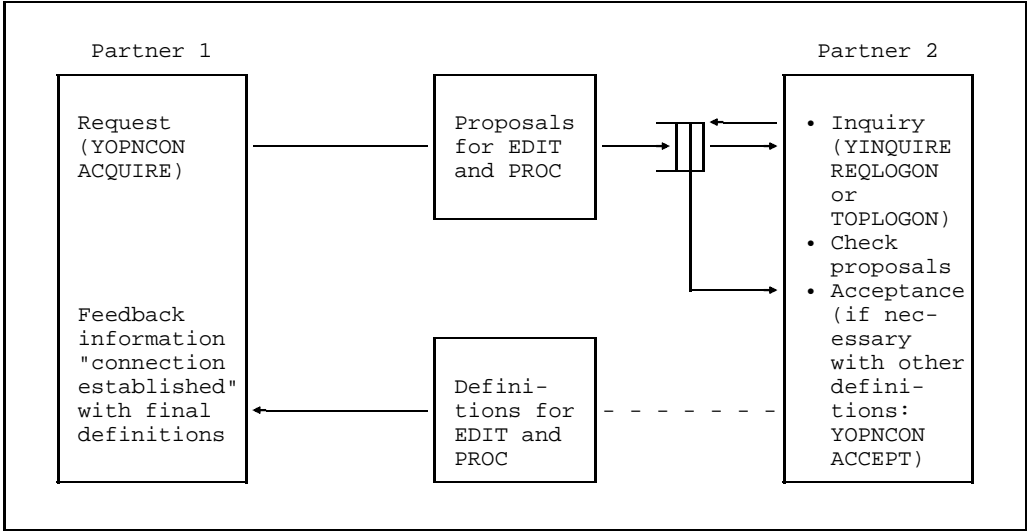
- By requesting the proposals from partner 1 (YINQUIRE) and then checking and either accepting or rejecting them, i.e. by sending other values for EDIT and PROC back to partner 1 (YOPNCON ACCEPT).
- By sending its own values for EDIT and PROC to partner 1 without considering the proposals.

In each case the values sent by partner 2 are binding for partner 1. This means the values suggested by partner 1 for EDIT and PROC need not necessarily correspond to the current values, and therefore the user must inform himself of the appropriate values after the connection is established.

Both in the case of a request for connection setup (YOPNCON ACQUIRE) and the acceptance of the connection (YOPNCON ACCEPT) the partners can send each other connection messages.

Note

If a connection was predefined, its characteristics were defined during system generation.



Agreement on characteristics

3.2.1.1 Definition of the connection to be established

The connection to be established has to be defined in the DCAM program. The following information is required:

– **Name and processor name (= address) of the partner**

This information is used to address a partner when a request is issued or a request is accepted by a specific partner (SPEC). Processor name = ' ' addresses the own processor as partner.

If an accept call was issued for any partner (ANY), DCAM returns the partner and processor names after completion of the call.

– **Accompanying information**

Here, the user defines an optional character string which, for example, is added to every message he receives via this connection. It may be the address of a data area to be assigned to this connection or the address of a routine which is to be activated specifically for this connection. This method is particularly useful for access to the common receiver queue with YRECEIVE ANY. The content of the accompanying information is optional; however, it may not exceed 4 bytes.

Restriction:

Accompanying information cannot be defined in COBOL.

– **Data flow control**

In the case of YSEND, there is a possibility of the connection being overloaded, for example because the partner - which may be in the same system fetches the notifications too slowly. If this happens the YSEND is not executed: instead, it is terminated with the RC "Wait for GO" (X'10040C00'). A subsequent YSEND on this connection cannot be successful until DCAM receives the GO signal. YSENDS issued before the GO signal is received are terminated with the RC "Shortage" (X'10040800').

If the connection was set up with PROC=SIGNAL, the user is notified when the GO signal is received (see SOLSIG, BS2000 Executive Macros). In COBOL, this takes the form of the arrival of the "GOSIGNAL" event after YWAIT. When the GO signal is received, the connection is free and data can again be sent. Note that when a connection is set up with PROC=SIGNAL, the YSEND must include the address of a valid EID.

If the connection is not set up with PROC=SIGNAL, a successful YSEND is the only way of ascertaining that the overload has ended.

The GO signal provides no guarantee that the next YSEND call will be successful.

If a connection is so overloaded that no messages can be forwarded, the user can issue a GO signal to find out when the connection will be free and data can be sent again (SIGNAL).

– **More-data function**

The MDATA=Y/N operand allows the application to specify whether it wishes to use the more-data function for the transfer of data units, (see page 34ff).

– **Length of the messages/data units**

In the MAXLN operand the application specifies the maximum message length (when MDATA=N) or length of the data units (when MDATA=Y) that are to be sent across this connection.

The actual length available (may be less than length requested by the application in the MAXLN operand) is supplied by DCAM as feedback information on connection setup (see pages 63 and 66).

This length is used to optimize the buffer made available by the system and is not passed on to, or negotiated with, the communication partner.

– **Length of expected received messages (with MDATA=N only)**

If the application is using the connection without the MDATA function (MDATA=N), it can tell the communication system what length it expects received messages to have via the RLTH operand. The size is required for memory space optimization and improving the system's performance. Its effect is strictly local and it is neither passed to, nor negotiated with, the communication partner.

The use of RLTH therefore offers no guarantee that even longer messages won't arrive and have to be processed by the application.

Note

In the case of MDATA=Y, DCAM supplies the maximum possible length of data units to be received as feedback information for connection setup. The application has no effect on the length that is fixed. The communication system ensures that no longer data units are passed on.

– Handling of long messages

When MDATA=N (no more-data function), even specification of the RLTH operand cannot prevent longer messages from arriving; DCAM offers two options here, depending on the particular problem (see table below).

This entry can be changed later on during the execution of a receive call (see page 83ff).

Requirement	Definition
<ul style="list-style-type: none"> - Only messages of a maximum fixed length are expected on this connection. This length should be specified in the RLTH operand. - A message that is longer than expected must be incorrect. The part which is overlong is therefore discarded. 	<p>The value TRUNC is specified for PROC.</p> <p>Messages that are longer than expected are only transferred up to the length expected. The overlong condition is indicated but discarded.</p>
<ul style="list-style-type: none"> - It is not possible to predict the length of the messages arriving via this connection. The length assumed to be most common is therefore specified and an input area reserved for it. - The possible excess length is not to be discarded; it will be fetched in a further receive call. 	<p>The value KEEP is specified for PROC. The overlong condition is flagged when the first part is received. The overlong portion is saved for another receive call.</p>

Note

As in the case of the more-data function (MDATA=Y) DCAM indicates the maximum length of the data units that are to be received, you should always reserve a receive area with this length. There will then be no danger of truncation and PROC=KEEP is irrelevant.

If you nevertheless select an input area which is smaller than the announced length for data units to be received, the table above applies analogously for each data unit of the message to be received.

– **Route selection**

Up to 8 (COBOL) or 16 routes (ASSEMBLER) can be specified to specific partners. DCAM tries to establish a connection via the routes specified in the order of their occurrence. When several routes have been specified, the application program cannot determine which one was used.



The rest of this section applies to DCAM(NEA) transport service applications only.

– **Terminal status**

In the case of terminals with status capability (e.g. 9763), the current terminal status can be requested on connection setup when virtual terminals are used. This completes the information collected by the VTSU on the terminals, e.g. on character sets loaded (see the VTSU User Guide).

– **Message code**

As different codes can be used on the transmission line sections in a data communication system, it is left to the user to select a code according to the job definition at hand; the table below provides an overview of the options.

Requirement	Definition
<ul style="list-style-type: none"> - The user is working with virtual terminals. - He wishes to receive messages in the code of his own host computer (as a rule in EBCDIC). - He always transmits messages in the code of his own host computer. 	<p>The value SYSCODE is specified for PROC. If necessary, the messages are converted in the data communication system.</p>
<ul style="list-style-type: none"> - The user is not working with virtual terminals. - He wishes to receive messages in the code used by the communication partner. - He transmits messages in the code which the partner can process. 	<p>The value BINARY is used for PROC. The messages are transmitted in any code (transparent).</p>

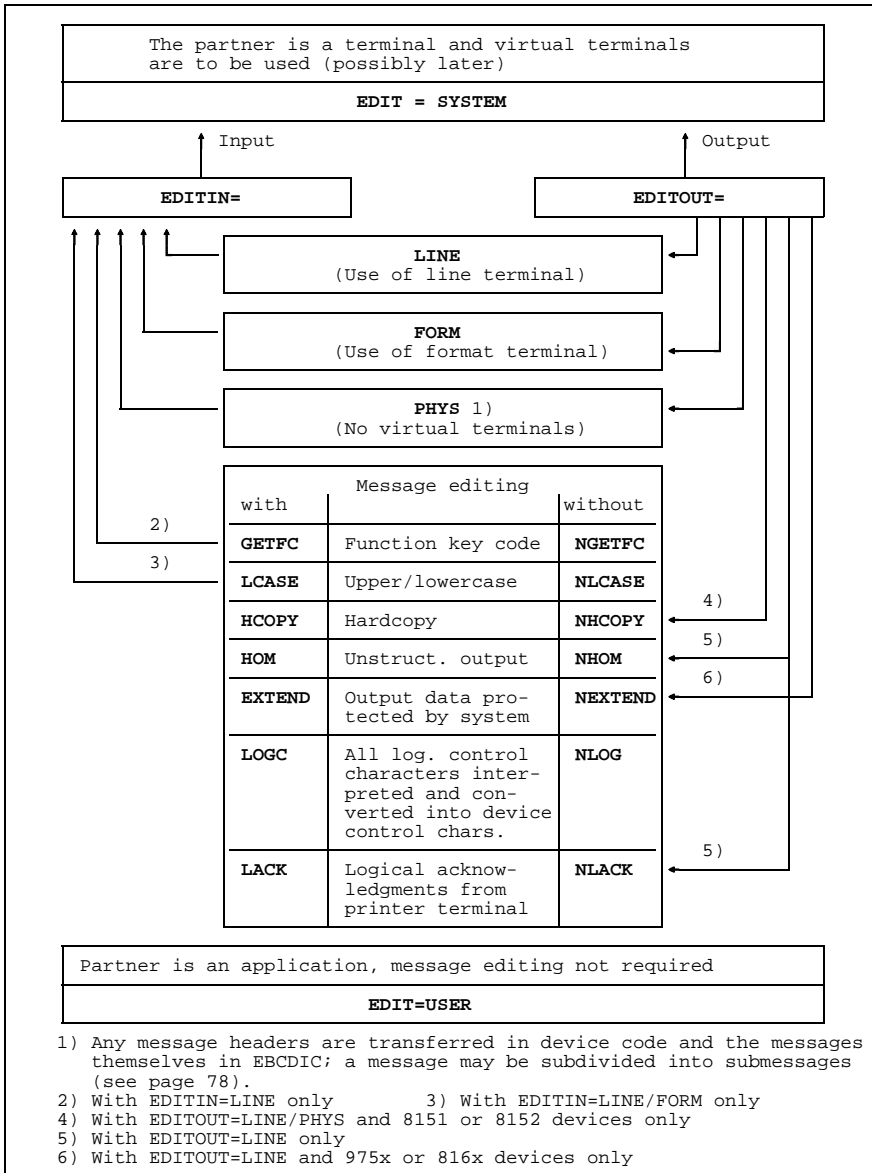
- The **logon password** is specified when a connection request is to be made. It must be specified in the same way as the partner (in this case a DCAM application) defined it on opening.
- Where **messages are distributed by means of distribution codes**, the reference to the detailed description of this distribution code must be specified here. This is described in detail on page 68.
- The **maximum length of data** (MAXLN) which is to be transmitted via this connection. This is a value ensuring optimum usage of the buffers provided by the system; it is not passed on to the communication partner.
- **Message characteristics which are agreed with the partner** As mentioned in the preceding section, the two partners must agree on some characteristics. These are also described here.

Initiation of data transmission is also defined. If the DCAM application is to initiate transmission, APPSTART is set. Otherwise, there is no definition (ANYSTART).

The type of **message editing** used is defined. It is possible to define message editing by means of the EDIT options (see figure below and page 95ff).

Note

For connections with EDIT=SYSTEM, where ATTR=DISCO the distribution code is always edited in the line mode.



Message editing options

3.2.1.2 Connection request

A connection request can be sent to **any partner** in the data communication system.



Applies to DCAM(NEA) transport service applications only:

If the partner is a terminal, the associated processor (e.g. a remote front-end processor) will answer the request. Otherwise, a response will come from the communication access method in BS2000 or from the partner (in this case a DCAM application). At present, there are two possibilities for the processor node to which the **terminal** is connected:

- If the processor node receives a request for a terminal that is not ready (switched off in the case of a dedicated circuit, busy in the case of a dial-up line, etc.), this will be rejected.
- If the terminal is ready, it is assumed that an operator is also present, and the connection is established (the processor node accepts the request).

The **communication access method** in BS2000 applies different criteria. The overview in the figure below shows that the DCAM user has several control options, including some in the program itself, during request processing. A connection message can be transferred along with the request for checking purposes or just for isolated transmission. This message can have any content and a maximum length of 32 bytes in the case of DCAM(ISO) or 80 bytes for DCAM(NEA). It is the only message that is transmitted without a connection already established.

A partner wishing to **issue a request** has to specify the following:

- Address of the partner (partner and processor name) to which the request is directed.
- A connection message (optional) to be transmitted along with the request.



Addition specifications for DCAM(NEA)transport service applications:

- Description of the connection characteristics
- Logon password, if required.
- Type of message distribution (if distribution codes are not used), i.e. whether the messages transmitted via this connection are to be distributed using the common receiver queue or the originator-oriented queue. This specification can be altered during data transmission.

If the request was accepted, the partner receives the following **feedback information**:

- The accompanying information defined by the partner.
- The maximum length of the messages/data units to be sent via this connection (MAXLN operand).
- The maximum length of the data units expected if the more-data function is used.



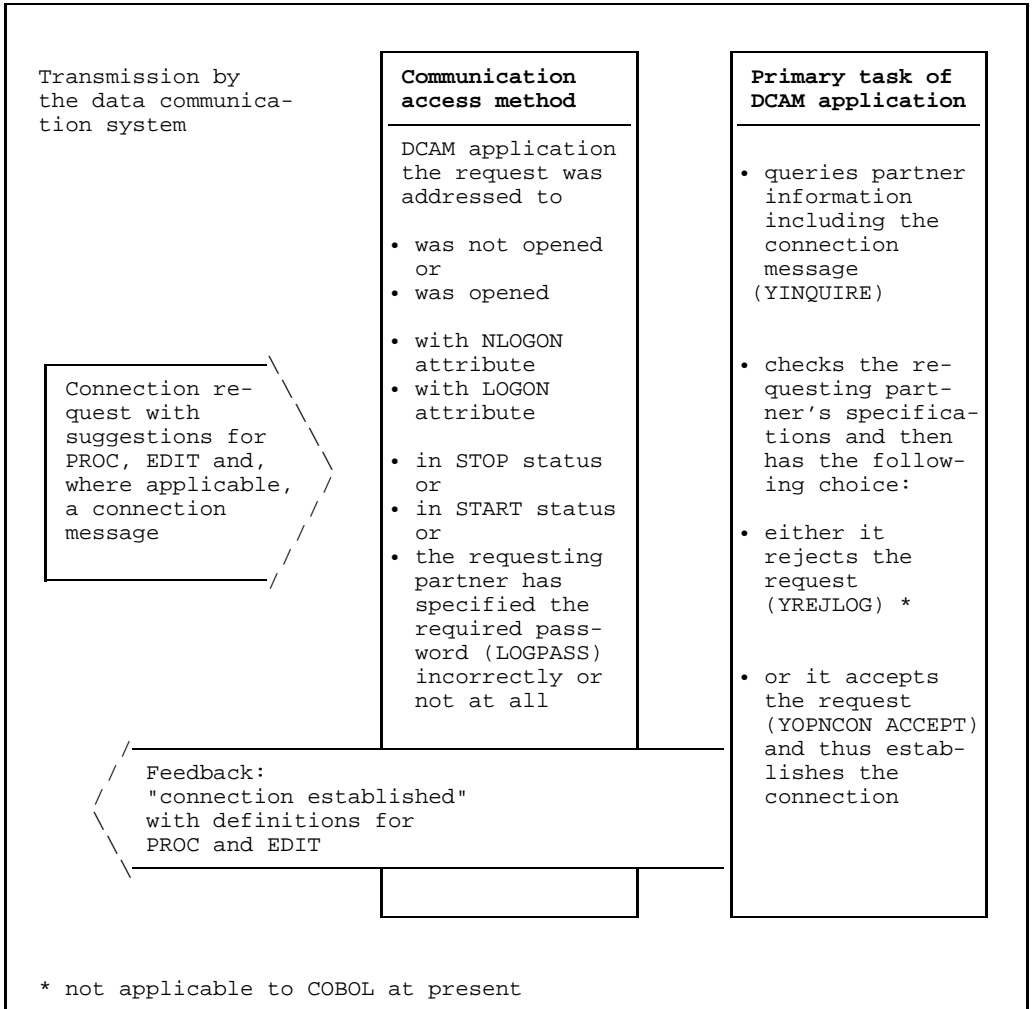
Additional information in the case of DCAM(NEA) transport service applications:

- The final definition for PROC and EDIT.
- Information on the partner (partner characteristics).

The possibility of having the statement contained in this call processed asynchronously is dealt with in the description of the language features.

Restriction on DCAM(NEA) transport service applications

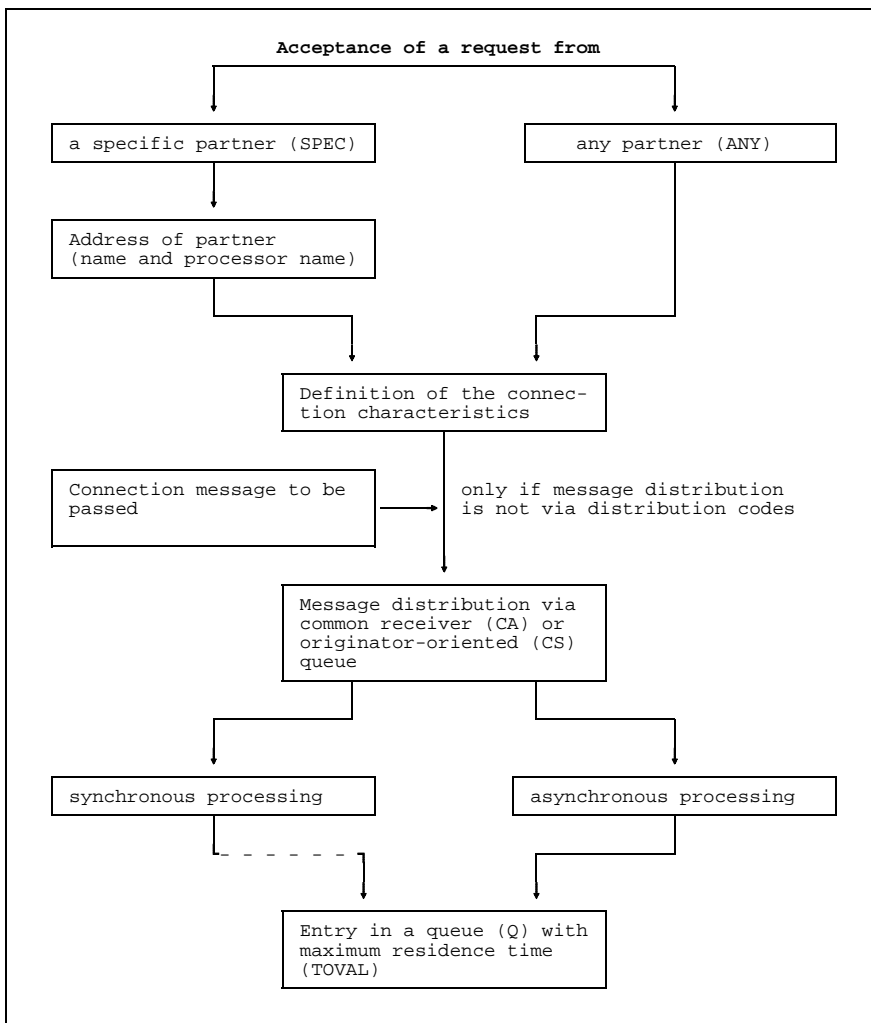
Connection messages are only delivered to DCAM applications. Terminals attached to Communication Computers cannot receive connection messages.



Processing a request for DCAM(NEA) transport service applications

3.2.1.3 Acceptance of a request

The basic description of acceptance is contained in the preceding sections. This section contains a summary of the information which can be specified for acceptance. The inter-relationship of the various specifications is shown in the figure below.



Acceptance of a request

Once the call has been executed, the task receives the following information:

- the maximum length of messages/data units which can be transmitted on this connection (see MAXLN operand)
- the maximum length of the data units that are expected if the more-data function is used



Additional information applying to DCAM(NEA) transport service applications only: information about the partner.

In addition, if the request accepted was that of any partner:

- the name of the partner and
- the partner's processor name
- accompanying information

3.2.1.4 Connection setup - use of distribution codes



This section applies to DCAM(NEA) transport service applications only.

If messages are to be distributed by means of distribution codes (defined when the application is opened), a number of definitions must be made when the connection is set up. The description of a connection contains, in this case, a reference to that section that describes the distribution codes. Distribution codes are described in 2 stages in order to provide maximum freedom in assigning connections to distribution codes.

Stage 1

Description of the code position in the message and of the length of the code(s) used. The code must be contained in the first 256 bytes of the message and can have a maximum length of 8 bytes.

In addition: reference to the description of the second stage, which may be present 8 times (COBOL) or 16 times (ASSEMBLER).

Stage 2

Description of the codes used by this connection. Up to 8 codes can be described.

These descriptions can be shared, i.e. several connections can use the same stage 1 description and several stage 1 descriptions can use a stage 2 description.

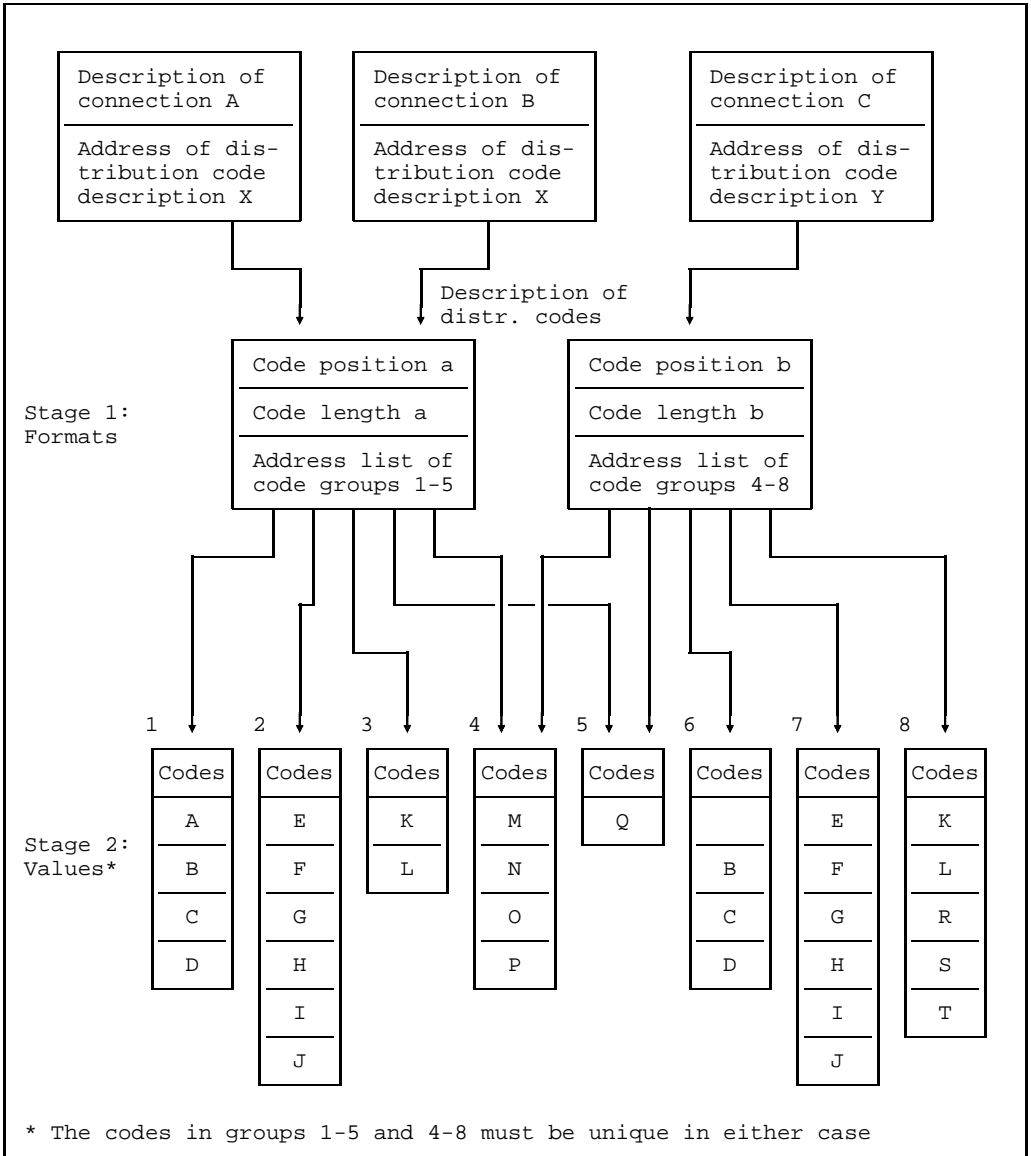
The following must be observed:

- The codes which are addressed by the same stage 1 description must be unique and
- must all be of the same length.
- If several stage 1 descriptions address a stage 2 description, the length specifications in the stage 1 descriptions must be identical.

The possibilities that result from this method of describing distribution code during the setting up of a connection are as follows:

- Code length, code position in the message and the code sign are defined when the connection is set up and cannot be altered.
- A partner can specify up to 64 (COBOL) or 128 (ASSEMBLER) different distribution codes in groups of 8. Thus, the partner can either reach one task via up to 8 different codes or he can read a varying number of tasks.

The number varies because the assignment of codes and task via the distribution code name takes place during execution. This assignment is controlled by the primary task by means of YPERMIT and YFORBID calls (see page 88ff).



Description of formats and values of distribution codes

3.2.1.5 Linking up to a predefined connection



This section applies to DCAM(NEA) transport service applications only.

A predefined connection is determined during the **generation of the communication access method**. It is established when the communication access method is started. However, the two partners must first connect themselves to the predefined connection before data transmission can begin.

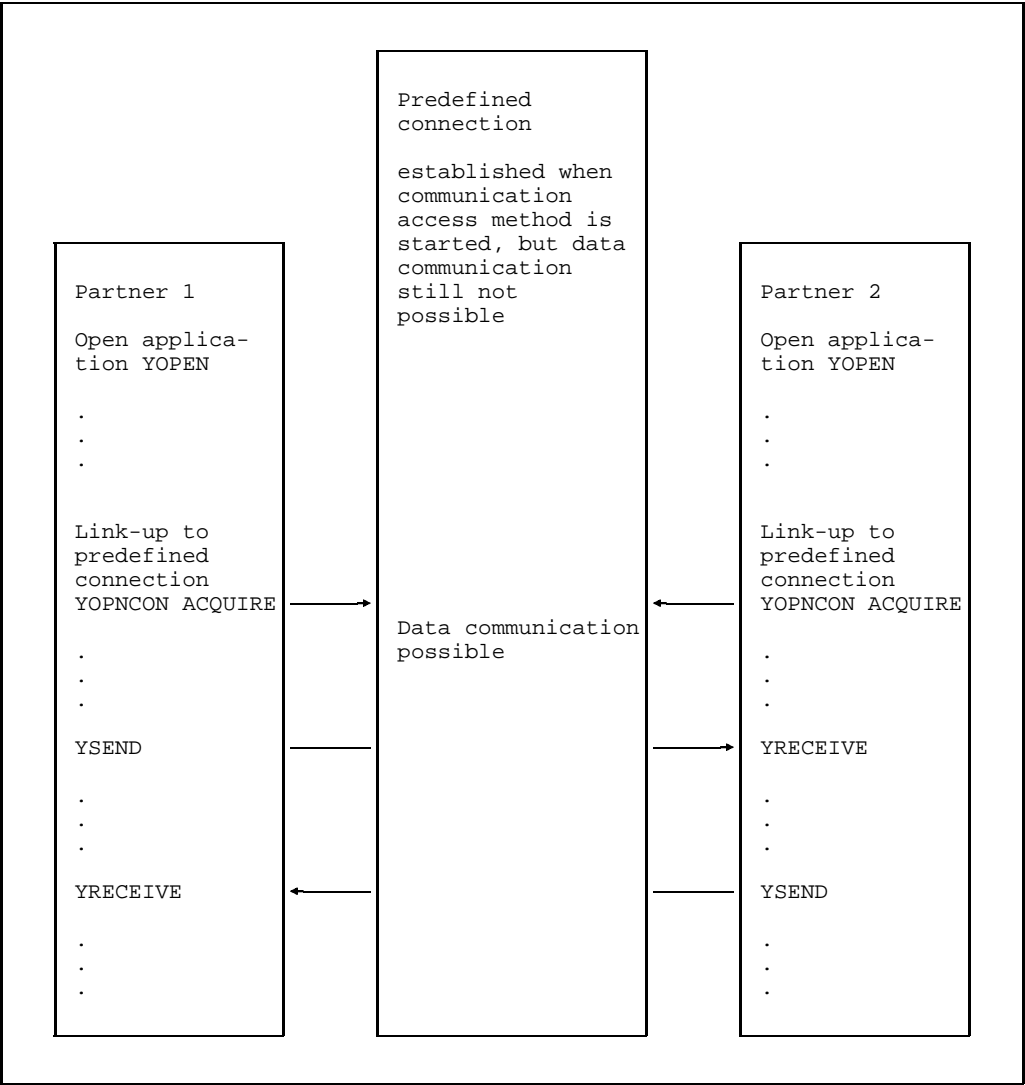
In the case of applications the primary tasks must make a connection request (YOPNCON ACQUIRE). In the case of a terminal this need only be activated and operational.

Because the characteristics of a predefined connection are determined during the generation of the communication access method, and because the connection is automatically established when this is started, certain **special factors** are involved:

- If data is transmitted before the partner has linked up with the connection, negative acknowledgments are given.
- If a partner wished to withdraw from a connection, the LOSCON routine is not initiated (the predefined connection remains established).
- A YOPNCON ACCEPT SPEC to a predefined partner can never be successfully terminated. If OPTCD=Q is specified, the call is entered in a queue and is terminated by timeout.

The following specifications cannot be made by a YOPNCON ACQUIRE if the connections are predefined:

- a connection message
- a LOGON password
- message editing by the system or the user (EDIT=SYSTEM/USER)
- start data transmission (PROC=APPSTART/ANYSTART).



Linking up to a predefined connection

3.2.2 Querying entries on partners and connections

As already mentioned in the sections on connection requests and the acceptance of requests, it is necessary to query the entries on partners and connections. The YINQUIRE call, of which several variants are available, is used for this purpose. For secondary tasks, only the CIDXLATE, NAMXLATE and, in the case of DCAM(NEA) transport service applications, the request for status information are applicable.

Restriction:

In COBOL not all variants of the YINQUIRE call are available.

The table below shows the YINQUIRE variants and the programming languages in which each can be used.

Inquiry	YINQUIRE variant	Programming language
<ul style="list-style-type: none"> - Which partner is requesting connection? - What connection message is to be transmitted and what is its length? <p>for DCAM(NEA) transport service applications only:</p> <ul style="list-style-type: none"> What are the proposed characteristics of the connection? 	Inquiry after an asynchronous LOGON notification (REQLOGON)	ASSEMBLER
<ul style="list-style-type: none"> - Which partner is next in the queue of those requesting connection? - What connection message is to be transmitted and what is its length? <p>for DCAM(NEA) transport service applications only:</p> <ul style="list-style-type: none"> What are the proposed characteristics of the connection? 	Inquiry prior to acceptance of a request (TOPLOGON)	ASSEMBLER/COBOL

Inquiry	YINQUIRE variant	Programming language
for DCAM(NEA) transport service applications only: What characteristic feature does the partner have?	Inquiry as to partner characteristics (PTNCHAR, MONCHARS, PEROTERM, BTERMINF)	ASSEMBLER/COBOL
<ul style="list-style-type: none"> - How many partners have sent a request? - With how many partners has a connection been established? 	Inquiry for number of partners (COUNTPTN)	ASSEMBLER/COBOL
<ul style="list-style-type: none"> - What is the identifier of the connection of which the partner and the processor names are known? 	Inquiry for identifier CID (NAMXLATE)	ASSEMBLER
<ul style="list-style-type: none"> - What are the partner and processor names of a connection whose identifier is known? 	Inquiry for the partner and processor names (CIDXLATE)	ASSEMBLER

3.2.3 Rejecting a connection request

The YREJLOG call can be used to reject a logon request; it suffices to specify the address of the requesting partner (partner and processor name) obtained by means of the YINQUIRE call.

3.2.4 Changing the characteristics of a connection



This section applies to DCAM(NEA) transport service applications only.

The user can change some of the connection characteristics defined during connection setup.

Only those characteristics can be changed which exclusively affect the partner making the changes, i.e. not those which have been agreed with the communication partner.

In the YCHANGE call, all of the values listed in the following must be set regardless of whether they are changed (new) values or the previously set (old) values.

The following can be changed:

- Accompanying information
- Handling of excess-length messages
- Code of the messages
- Values for VTSUCB (message editing) see the VTSU User Guide.
- If required, address of the distribution code description for changing distribution codes described in groups (the length and the position in the message and the code sign cannot be changed).

For further information on these values see "Definition of the connection to be established" (see page 57).

3.2.5 Deleting a connection request

A connection request sent to a partner can be deleted with the YCLSCON call.

The connection request is revoked if the connection is not yet established. If already established it is cleared down.

3.2.6 Closing a connection

Explicit closing of a connection is possible with the YCLSCON call.

There is no obligation to do this, as all connections are implicitly closed when the DCAM application is closed. Uncollected messages are destroyed when the connection is closed. Any messages sent immediately before YCLSCON, which had not yet been transmitted to the receiver, may also be destroyed.



In the case of DCAM(NEA) transport service applications it is advisable to save the final message with a transport acknowledgment.

Explicit closing may be required for:

- putting a time limit on a connection
- regulation of data throughput (connections can be closed down in order to give other connections better throughput rates), etc.

3.3 Data communication function

The requirements for data transmission are fulfilled when a DCAM application has been opened and a connection established.

The data transmission function performs the following operations:

– **Transmit a message/data unit (YSEND)**

The message/data unit to be transmitted to a communication partner is transferred to the data area of the communication access system.

– **Receive a message/data unit (YRECEIVE)**

The message/data unit from a communication partner (any one, a specific one or, in the case of DCAM(NEA) transport service applications, one with a specific distribution code) is entered in the data area of the user program.

– **Transmit and receive combined (YSENDREC)**

A message/data unit is entered in the data area of the communication access method by means of this call, and a message/data unit from the same communication partner is entered in the partner's own data area.

– **Delete receive calls and change the CS/CA state of a connection (YRESET)**

With this call a task can delete all its waiting YRECEIVE calls from a specified application or a specified connection, and it can alter the CS/CA state.



The following additional options are available in the case of DCAM(NEA) transport service applications for controlling message distribution by means of distribution codes:

– **Assign distribution code names to distribution code groups (YPERMIT)**

The primary task assigns specific distribution codes to the distribution code name of one or more secondary tasks.

– **Cancel assignment (YFORBID)**

An assignment is canceled by the primary task.

3.3.1 Sending a message/data unit

Every task in a DCAM application can transmit a message/data unit (YSEND) and this results in the following action:


- The data to be sent to a partner in the form of a message/data unit is transferred to the data area of the communication access method.
- The access method is assigned the job of transmitting the data to the partner.
- The call is terminated and the access method starts transmission.

The access method receives a number of user specifications along with the message/data unit to be transmitted. They are supplemented by specifications defined when the connection was set up (see page 57).

The following table lists any additional specifications required under certain circumstances.

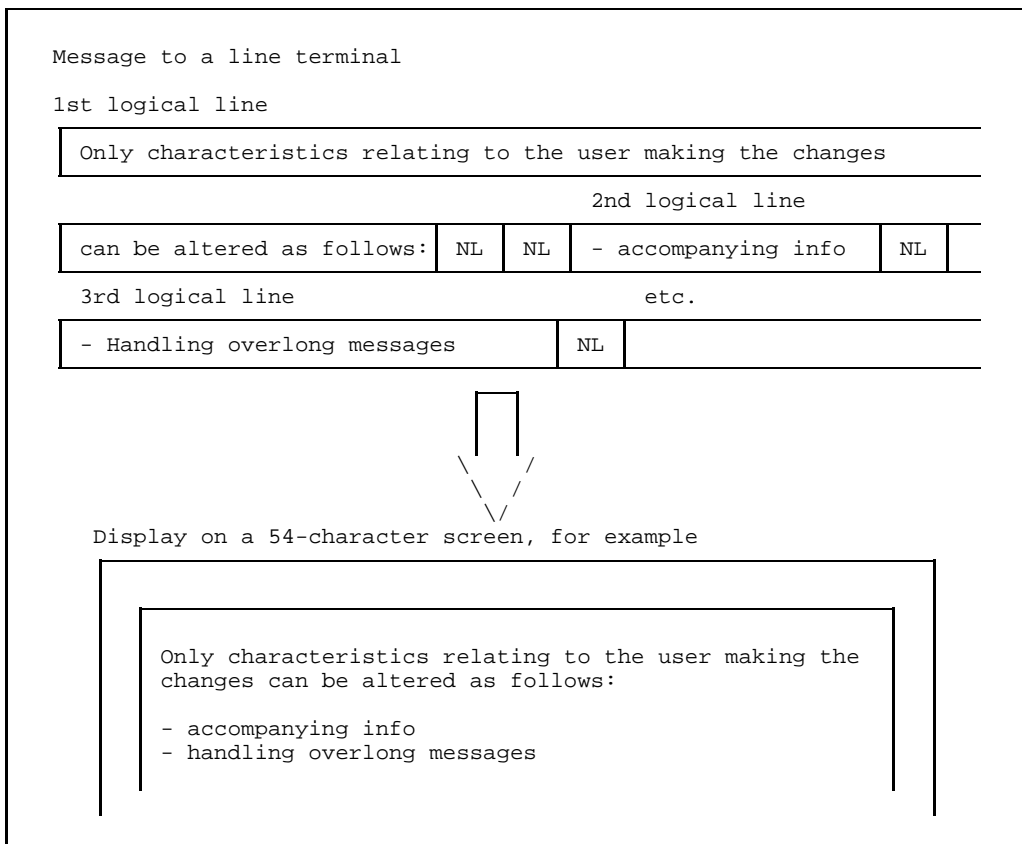
Requirement	Definition
After logon, the system announced the maximum length of messages/data units to be transmitted via this connection.	The length of the message,data unit to be transmitted is specified within the maximum length limits.
Messages/data units arriving on this connection are to be received via the originator-oriented queue.	The originator-oriented queue is set (CS state of the connection).
Messages/data units arriving on this connection are to be received via the common receiver.	The common receiver is set (CA state of the connection).
Upon opening of the connection PROC= SIGNAL was used to request the GO signal in the event of the connection being overloaded.	For each YSEND call a valid event identifier (EID) should be specified, via which a GO signal can be issued if necessary.
The more-data function is being used across this connection (MDATA=Y).	The user must specify whether the data unit is (OPTCD=GROUP) the last one of a logical message or not (OPTCD=ELEMENT).

Requirement	Definition
<p>The following applies to DCAM(NEA) transport service applications only:</p> <ul style="list-style-type: none"> - Acknowledgment reception was enabled when the application was opened (for the primary task = PRIMTASK for the requesting task = REQTASK). - Transport acknowledgments are to be processed. - The YSEND call should be followed immediately by a YCLSCON or YCLOSE call which means the connection may be closed before the data has been transmitted. The transport acknowledgment should always be awaited in this case. 	<p>A sequence number is specified so that the transport acknowledgment can be assigned to the correct message (maximum value=2047) when it arrives.</p> <p>A positive acknowledgment is requested (negative acknowledgments are generated automatically if the message cannot be delivered).</p>
<p>Message editing was not provided for by the system on this connection. The following was set:</p> <ul style="list-style-type: none"> - EDIT=USER: The user edits the messages in such a fashion that the communication partner can process them. - EDIT=SYSTEM and EDITOUT=PHYS: The system performs blocking for devices with a small input buffer. Everything else is provided by the user in such a way that the partner can process it. 	<p>The message can be classified as follows:</p> <ul style="list-style-type: none"> - ELEMENT: - SUBGROUP: - GROUP: <p>This identification of the message is transferred by the network to the receiver, which receives it along with the message. If the receiver is a terminal, an element is transmitted as a message segment, while a subgroup or a group is transmitted as a message followed by transmission termination.</p>
<p>Message editing is provided by the system on this connection.</p>	<p>Special rules apply to the use of virtual terminals (see the VTSU User Guide).</p>
<p>A special situation has arisen (e.g. backlog on the connection), which was notified by the feedback information for a rejected YSEND. The partner is to be requested to issue receive calls immediately.</p>	<p>The message is declared as an express message, which means that its length may not exceed 8 bytes.</p>

 The rest of this section applies to DCAM(NEA) transport service applications only.

The message layout has to be varied in accordance with the **message editing** option selected:

When a **line terminal** is used (see the VTSU User Guide), the user can subdivide the messages into logical lines. Each of these lines is terminated with the "new line" character, hex 15. The text following the NL character is automatically placed at the beginning of the next line by the terminal. If the logical line length exceeds the line length of the terminal, the logical line is subdivided automatically. The beginning of a message is always placed at the beginning of a line. This is illustrated below by means of an example.

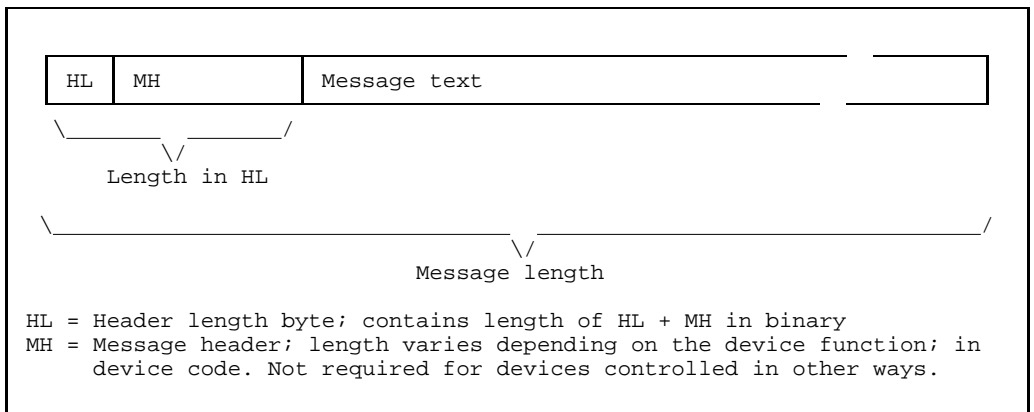


Message to line terminal

When a **format terminal** is used, the user transfers the message in the way he or she receives it from the formatting routine, but without the preceding record length field. The record length must be transferred from the record length field to the corresponding DCAM field.

If the values **EDIT=SYSTEM** and **EDITOUT=PHYS** have been set for the connection, each message must be preceded by a header length byte. The binary number contained in this byte specifies the length of the following message header (including the length byte). The message header is transferred to the terminal without code conversion in contrast to the message proper, so that the user can use the device description during generation and analysis of the message header. For devices which do not process message headers, the message must nonetheless be preceded by a header length byte, and the length must be set to 1.

The diagram below illustrates how messages should be formatted.



Message layout for EDITOUT=PHYS

The message can also be subdivided into message segments in this case and also when EDIT=USER has been set. Each message segment must be appropriately marked.

Message group						
Message subgroup 1				Message subgroup 2		
Message 1	Message 2	Message 3	Message 4	Message 5	Message 6	Message 7
ELEMENT	ELEMENT	ELEMENT	SUBGROUP	ELEMENT	ELEMENT	GROUP

Example of message structuring

Depending on the device type and status, different device responses have to be reckoned with.

Note

Shared physical lines (concentrator) are released once a message segment has been transmitted.

3.3.2 Receiving a message/data unit

Reception of a message/data unit means that the message/data unit from the partner is entered in the data area of the receiving partner's program after having been removed from the queue.

This procedure is completed when the YRECEIVE call is terminated provided that the call was executed synchronously. If the call was executed asynchronously, its completion merely means the communication access system has been assigned the job of transferring the message/data unit when it arrives. An agreed event identifier enables the user to receive a signal when the message/data unit been transferred. The user can check for this signal and then process the message/data unit (see page 105). This method allows any delays arising from waiting for a message/data unit to be used for other processing operations.

Which messages/data unit a task can receive depends on the type of queue selected:

- If the common receiver queue is set (CA state of the connection), the message/data unit is passed to the first task to access the common receiver queue (ANY).
- If the originator-oriented queue is set (CS state of the connection), the message/data unit is passed to the task that set the originator-oriented queue (causing the CS state).



Which message a task can receive also depends in the case of DCAM(NEA) transport service applications on whether it is a normal or an express message. The table below illustrates message distribution according to the type of message.

Type of message distribution		Type of message	
		Normal message	Express message
By means of distribution codes (DISCO)		Distributed to the task authorized to receive it according to distribution code 1)	Distributed to the primary task
Without distribution codes (NDISCO)	Via common receiver queue (connection state CA)	Distributed to the first task to access the common-receiver queue (ANY)	Distributed to the primary task
	Via originator-oriented queue (connection state CS)	Distributed to the task that set the originator-oriented queue (caused the CS state)	

1) If the distribution code has not yet been provided for or has not been assigned to any secondary task, the message is distributed to the primary task.

When issuing the YRECEIVE call, the user specifies further information beyond the connection description defined with YOPNCON (see page 57):

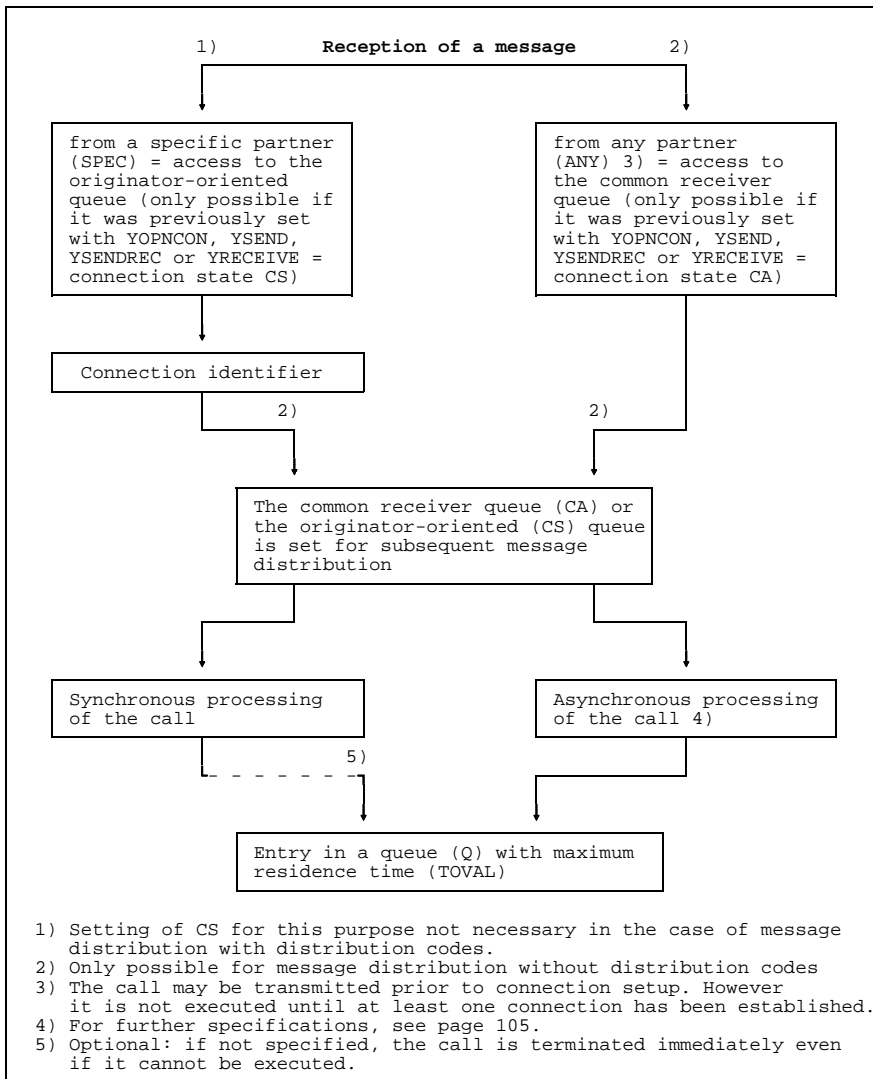
– With respect to the **message/data unit**:

the user defines the address of the field in which the message/data unit is to be entered, along with the length of this field;

he can change the handling of excess-length messages defined during connection establishment to match this message; i.e. he can redefine whether the excess portion of a message is to be kept so that it can be fetched later or whether it is to be discarded.

- With respect to **call execution**:

interrelated information must be specified here, as illustrated below:



Specifications for receiving a message/data unit

After the message has been received in the specified area, further information is fed back by DCAM:

- **Originator specification**,
if the message was received from any partner (ANY).
- **Accompanying user information**
as defined by the user when the connection was established (message received from any partner: ANY).
- **Length** of the message/data unit
or, if the message/data unit is longer than the storage area, provided, the length of the excess portion not yet transferred (KEEP). If the excess-length part is to be fetched with another receive call, the originator-oriented queue must have been selected previously.

When the more-data function (MDATA=Y) is used, an indicator in the feedback field is set showing whether the data unit is the last unit of a logical message.



The rest of this section applies to DCAM(NEA) transport service applications only.

- **Sequence number** of message
as specified by the partner or generated by the system (if a terminal was the originator).
- **Structure** of the message
(entries in the feedback field), i.e. whether an element, the last element of a subgroup or a group was received.
- **Type** of message
(entries in feedback field), i.e. whether a normal or express message was received.

Whether or not **transport acknowledgments** can be received depends on the definitions made when the application was opened and on the information specified when the message which is to be acknowledged is transmitted (see table below).

Definition at YOPEN of the application in the primary task	Definition for YSEND or YSENDREC	
	A positive acknowledgment is requested (TACK)	A positive acknowledgment is not requested (NTACK)
The requesting task is to receive acknowledgments (REQTASK)	Transfer of positive and negative acknowledgments to the requesting task	Transfer of negative acknowledgments only to the requesting task
The primary task is to receive all the acknowledgments (PRIMTASK)	Transfer of positive and negative acknowledgments to the primary task	Transfer of negative acknowledgments only to the primary task
No acknowledgments are to be passed (NOTACK)	Neither positive nor negative acknowledgments are passed	

Unless the use of asynchronous notifications was agreed (see page 111), transport acknowledgments are received with the YRECEIVE call. All information on the type of acknowledgment etc. is provided in the feedback field of the call. The field 'TACKNO' also contains the sequence number of the message as defined during transmission. It is thus possible to identify the message.

It should be noted that the reception of a transport acknowledgment with YRECEIVE has no effect on the setting of the queue (CS/CA) even if this was specified in the call.

Restriction:

Where EDIT=SYSTEM, a second negative transport acknowledgment can be received when YSEND is specified. This is passed to the primary task in the case of SHARE applications irrespective of REQTASK.

3.3.3 Combined sending and receiving

The information given on the two separate calls also applies to the combined call. As the combination only permits access to the originator-oriented queue, the originator-oriented queue must have been set for this connection (=connection state CS) with YOPNCON or a previous YSEND or YRECEIVE.

Restriction:

The YSENDREC call is not available in COBOL.

3.3.4 Canceling receive macros and changing the connection state CS/CA

The YRESET call enables asynchronous YRECEIVE macros to be canceled.

A **YRESET(ANY)** call cancels all YRECEIVE macros from any partner.

A **YRESET(SPEC)** call cancels YRECEIVE calls from a specified partner.

A YRESET(SPEC) call also resets the connection state CS/CA. This does not apply to DCAM(NEA) transport service applications using distribution codes specific to queues (ATTR=SHARE,DISCO).

3.3.5 Control distribution code assignment

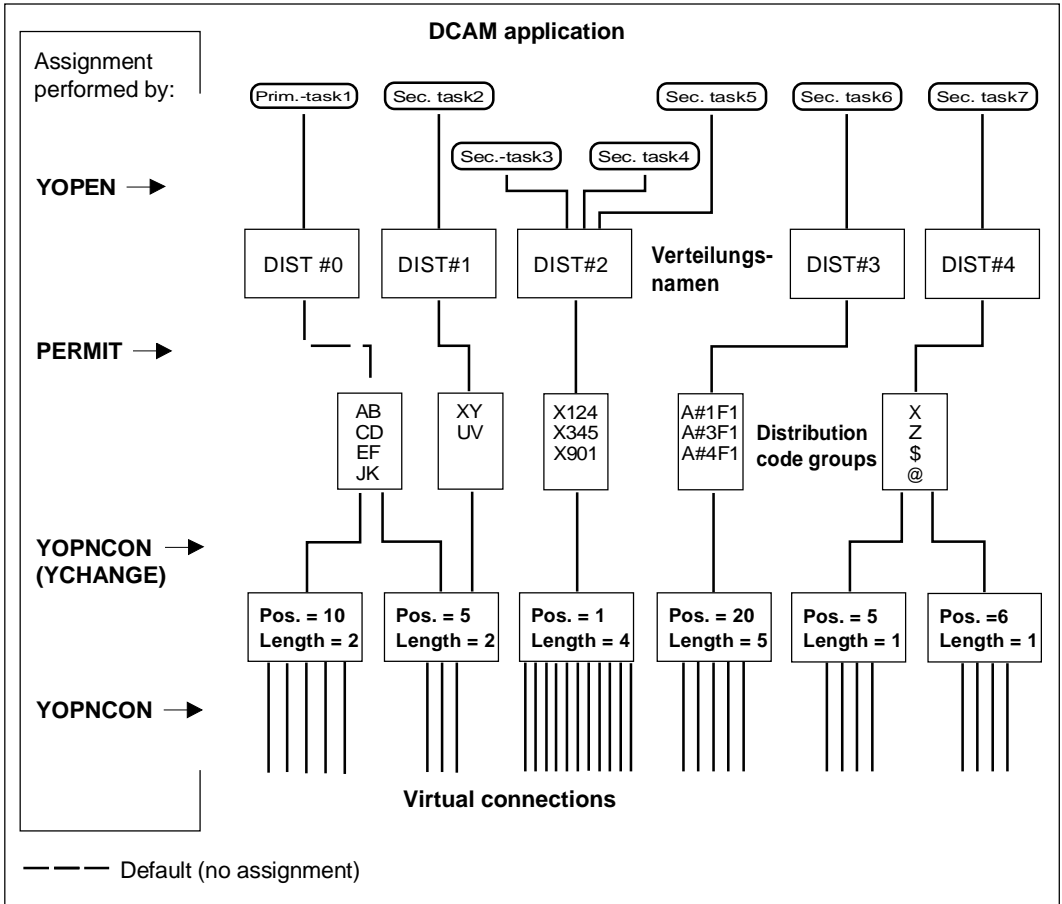


This section applies to DCAM(NEA) transport service applications only.

The primary task controls distribution code assignment by allocating a distribution code name to a distribution code group or by deleting such an assignment. It defines the distribution code group for one or more connections when establishing connections (see page 68).

The distribution code name is entered by one or more tasks when opening the application (see pages 42 and 47).

The primary task assigns the distribution code names of one or more tasks to the distribution codes of one or more connections. If the primary task deletes such an assignment, it inhibits data transfer from the connections concerned to the task concerned.



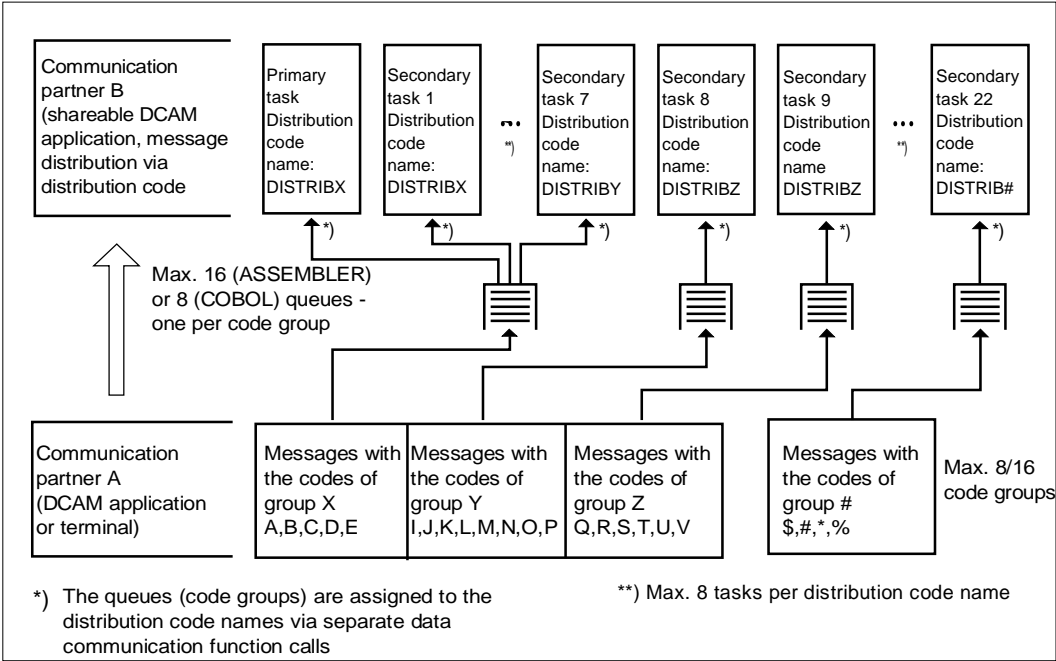
Summary of message distribution with distribution codes (example)

Assign distribution code name to a distribution code group

The YPERMIT call assigns a distribution code name to a distribution code group. This means that tasks which have defined this name can receive messages with the codes of the assigned group.

The following applies:

- A group of distribution codes (stage 2 definition) can be assigned to only **one distribution code name**.
- Up to **8 tasks** can use **the same distribution code name**. This becomes necessary when a single program controls several tasks (SHARED CODE, see pages 115 and 125). Tasks with identical distribution code names are served in accordance with the FIFO principle (FIFO=first in first out: the first entry in the queue will be the first one processed).
- Depending on the assignment made, a task can also access the messages in an **originator-oriented** way (YRECEIVE SPEC). It is not necessary to select this queue beforehand (there is no CS/CA state of the connection when distribution codes are used).



Example of task group addressing with distribution codes

Messages with distribution codes that were not assigned to a distribution name or that cannot be interpreted are directed to the primary task.

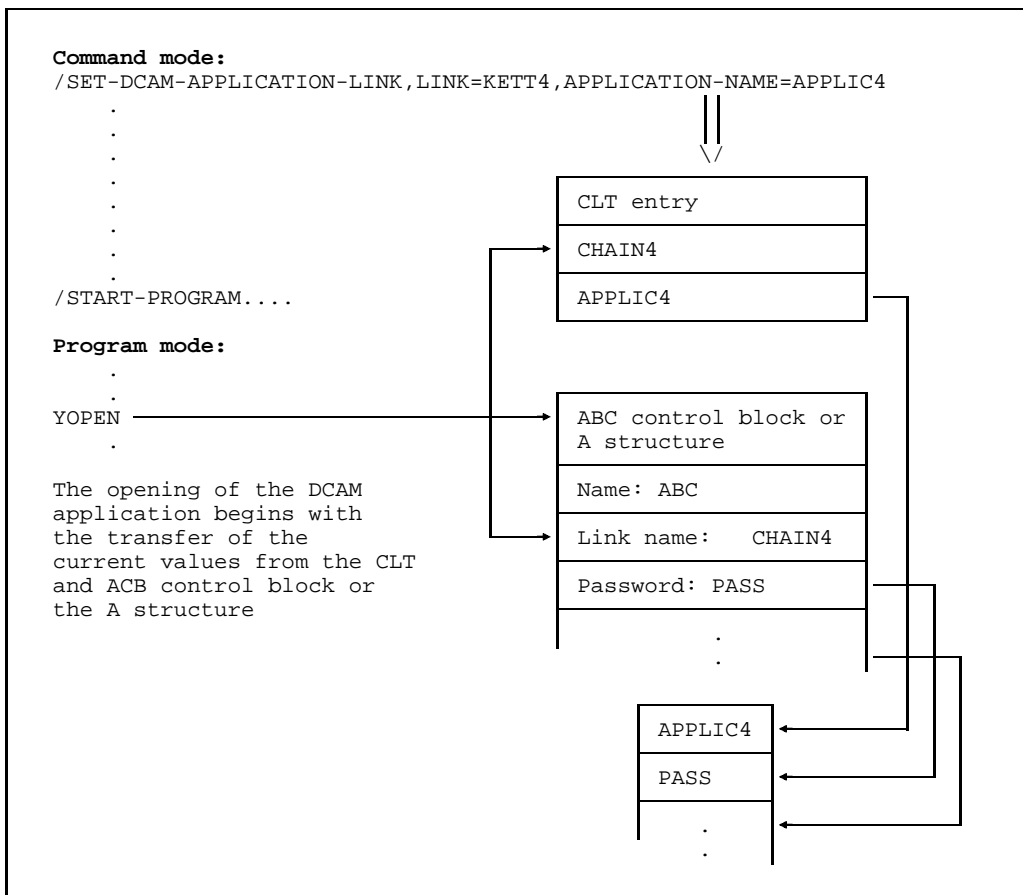
Old assignments can be replaced by new ones when a new YPERMIT call is issued.

Cancel assignment

If a new assignment is not to be made for a distribution code name but the existing assignment is to be canceled, the YFORBID call can be used. Data which has already been received is then transferred to the primary task.

3.4 Name assignment function

This function permits parameter values for the DCAM application or the connection to be specified **at run-time only**. This is achieved by linking the current values with those specified in the program in a task-oriented table (CLT=communication link table). (This can be compared with the entries in the FCB (file control block) of a program which are updated using the TFT (task file table) whose entries were generated by the FILE macro or the /SET-FILE-LINK command.)



Example of assignment of a DCAM application name

The table is set up:

- in the program mode (ASSEMBLER only) by means of the YAPPL macro for entries on the application, and the YCONN macro for entries on the connection
- in the command mode (ASSEMBLER/COBOL) by means of the
 - `/SET-DCAM-APPLICATION-LINK` or `/REMOVE-DCAM-APPLICATION-LINK` for entries on the application and
 - `/SET-DCAM-CONNECTION-LINK` or `/REMOVE-DCAM-CONNECTION-LINK` for entries on the connection (see "User Commands (SDF Format)").

The linkage is established by specifying a **link name** both in the program and in the CLT. The following values can be updated in this fashion:

- **For a DCAM application:**

- the name of the DCAM application
- the password for the connection of a secondary task to an application, as specified in the primary task
- the password for the connection of a secondary task to an application, as specified in the secondary task.



Additional information for a DCAM(NEA) transport service application:

- the distribution code name;
- the password for establishing a connection, as defined in the primary task.

- **For the connection:**

- the name of the partner;
- the name of the processor node to which the partner is connected
- the accompanying information.



Additional information for DCAM(NEA) transport service applications: the password for connection establishment, as specified by the requesting task.

When YOPEN or YOPNCON is executed, the specifications are entered in DCAM control blocks or data structures. They are updated with the values in the CLT providing it contains entries. If it does not, the original values remain unchanged. Following this, dynamic name assignment for this application or connection is possible again only after it has been closed. If CLT entries are to be deleted, the YAPPL or YCONN macro or the corresponding commands with just the link name should be issued.

4 Support for virtual terminals



This chapter applies to DCAM(NEA) transport service applications only.

A virtual terminal constitutes a model of a terminal with certain standard characteristics. Unlike a physical terminal, the virtual terminal permits programming that does not relate to the physical characteristics of the terminal. Actually connected. The physical terminal on which he or she wants to output messages is therefore of no significance to the user, who is aware only of the virtual terminal.

There are two types of virtual terminal:

- form terminal and
- virtual line terminal

Form terminal

The forms (screen forms or masks) are output on the form terminal. The user completes only the predefined fields. The form terminal offers the program structured forms with a field structure. Form terminals are supported by restricted forms mode, which is selectable with VTSU, and by the software product FHS (Forms Handling System).

Detailed information on FHS is available in the FHS User Guide.

Virtual line terminal

The virtual line terminal is implemented by the software product VTSU (Virtual Terminal Support) in the BS2000 operating system. In the case of a virtual line terminal an output is divided into a number of lines. VTSU supports both Assembler macros and COBOL data structures for line terminals. By using these macros and data structures you can:

- create receive fields or symbolic field names for status messages. You can then query the corresponding status messages with the YINQUIRE macro.
- Convert virtual control characters into device-specific control characters. The mechanisms for this purpose are the VTCSET macro and the TIAMCTRC copy element.
- Set VTSU parameters. These VTSU parameters define the nature of the virtual terminal and how messages are treated for send and receive. You can specify the parameters in the VTSU control block (VTSUCB) in the YSEND, YRECEIVE and YSENDREC macros.

VTSU is described in detail in the VTSU User Guide.

The status messages on the partner characteristics for COBOL applications are described in the 'DCAM COBOL Calls' manual.

Edit options

The edit options for connections with EDIT=SYSTEM are replaced by the specifications in the VTSUCB. As in the VTSUCB, you can use edit options to define the type of virtual terminal and message handling for transmitting and receiving. The functionality of the VTSUCB is considerably wider, however, because new edit options will be available only via the VTSUCB.

5 DCAM programs

This chapter describes the language-specific characteristics of the DCAM interface. DCAM programs can be generated in ASSEMBLER or COBOL. For detailed instructions on the coding of DCAM programs consult the manuals "DCAM Macros" for ASSEMBLER and "DCAM COBOL Calls" for COBOL.

5.1 ASSEMBLER programs

The ASSEMBLER program interface provides the full range of DCAM functions as described in detail above.

The purpose of this section is to provide an overview of the specific interface characteristics.

5.1.1 Macro calls and control blocks

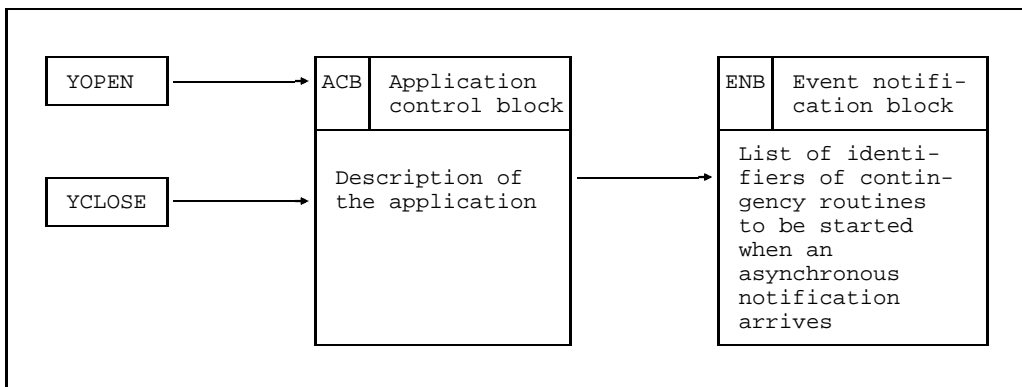
All DCAM functions are activated by macro calls. The action macros always refer to control blocks in which all the parameters are stored. A distinction is made between two groups of calls depending on the type of control block they refer to:

- macro calls that refer to an **ACB (application control block)**, and
- macro calls that refer to an **RPB (request parameter block)**.

The following calls refer to an **ACB**, which contains all the information on the DCAM application:

- **YOPEN** (Open a DCAM application)
- **YCLOSE** (Close a DCAM application)

If asynchronous DCAM notifications are to be processed, an **ENB (event notification block)** is required in addition. The references to the identifiers for the contingency routines are stored in this block (see page 111).



ACB-related macro calls

The following macros refer to an **RPB** which contains the current parameter values of the action calls:

- **YINQUIRE** Request entries on DCAM applications, partners and connections
- **YOPNCON** Open connection
- **YCLSCON** Close connection
- **YREJLOG** Reject logon request
- **YCHANGE** Change connection characteristics
- **YSEND** Transmit message
- **YRECEIVE** Receive message
- **YSENDREC** Transmit message and receive new message
- **YRESET** Delete receive messages and change CS/CA state

! Additional macros for DCAM(NEA) transport service applications:

- **YSETLOG** Change status of DCAM application
- **YPERMIT** Permit receipt of message with distribution code
- **YFORBID** Prohibit receipt of message with distribution code

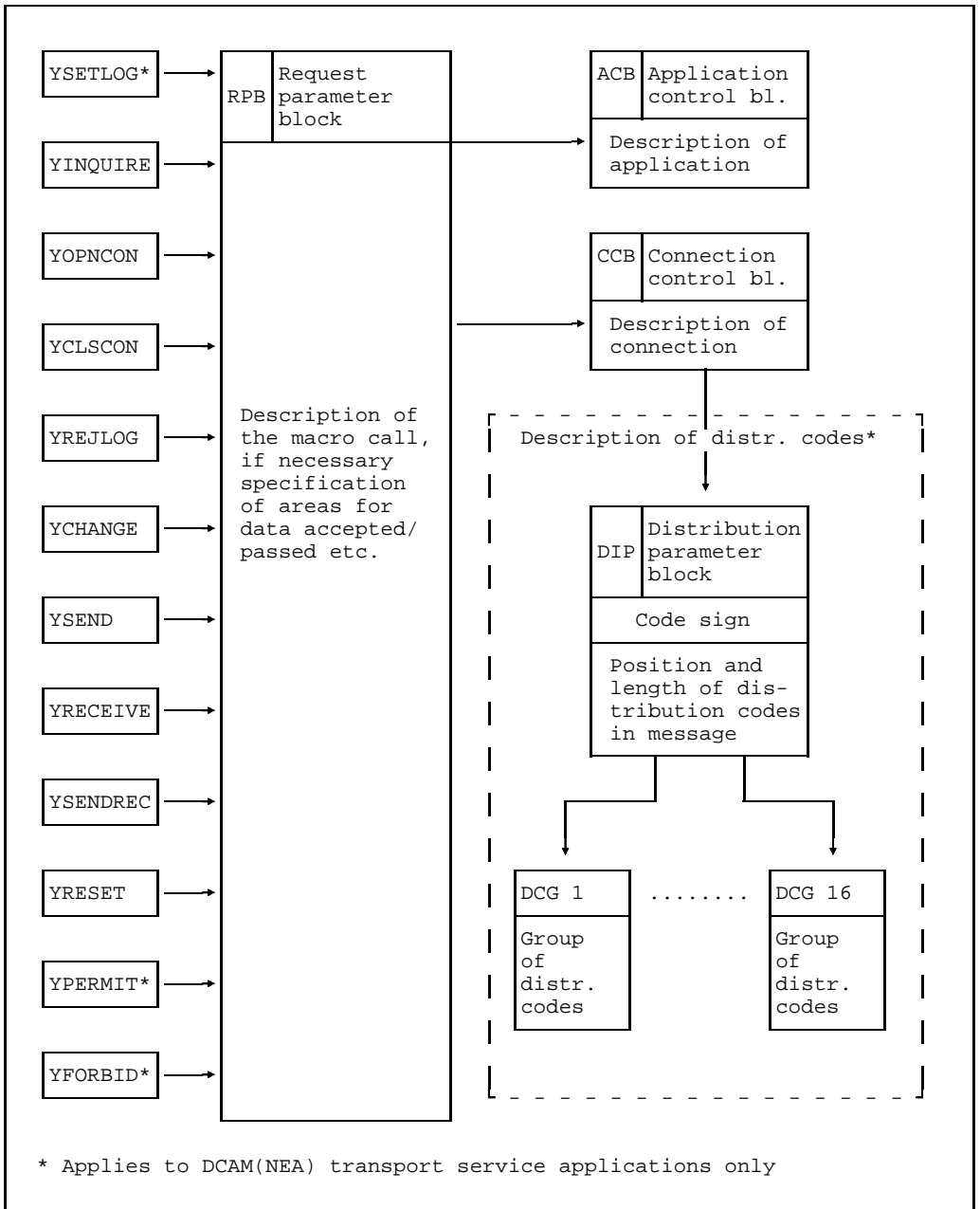
The **connection control block (CCB)** is required in addition to the RPB for various calls. The parameters for a connection are located in the CCB.



In the case of DCAM(NEA) transport service applications, the position and length of the distribution code, the code sign and the references to assigned distribution code groups are located in the **distribution parameter block (DIP)**, and the distribution codes themselves in the **distribution code group block (DCG)**.

Note

DCAM users connected to a packet switching network (e.g. Datex-P) via an X.25 interface must note that certain restrictions apply to calls relating to connections or data transmission. They are described in the manual "DCAM Macros", see section entitled 'Effects of the CCITT X.25 Recommendations on the IDCAM user interface'.




RPB-related macro calls

The programmer need not know the internal structure of the control blocks as macro calls are available to him for the generation and handling of control blocks.

Macro calls for **static control block generation**:


- **YACB** Generate ACB
- **YENB** Generate ENB
- **YRPB** Generate RPB
- **YCCB** Generate CCB

 Additional macros for DCAM(NEA) transport service applications:

- **YDIP** Generate DIP
- **YDCG** Generate DCG

In the case of these macro calls, the control blocks are generated during assembly.

The control blocks can also be generated and handled by the user directly via the macro calls YDDACB, YDDCCB, YDDENB and YDDRPB,

 and also with YDDDCG and YDDDIP in the case of DCAM(NEA) transport service applications.

These macro calls describe the layout (DSECT and CSECT) of the control blocks.

Dynamic generation of a control block in user memory area (class 6 memory) or optionally in an area outside the user program (class 5 memory) during the program run is possible with the macro call **YGENCB** (generate control block).

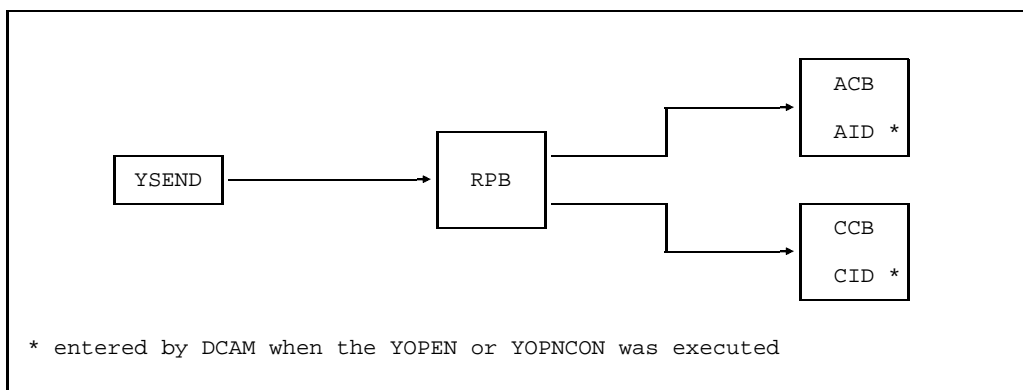
The following macros are provided for **control block handling**:

- **YMODCB** Modify the contents of control block fields
- **YSHOWCB** Transfer the contents of identified control block fields to an area reserved in the program
- **YTESTCB** Compare the control block field contents with specified values.

The control blocks can be created and handled with the aid of these macros only when the DCAM subsystem has been loaded successfully. Note, too, that the DCAM subsystem status cannot be HOLD/DELETE when any of these macros is used. If a task successfully issued a DCAM command or a DCAM call before entering HOLD/DELETE, it can work with DCAM until the task is ended, despite a /HOLD subsystem or /DELETE subsystem (also applicable to %).

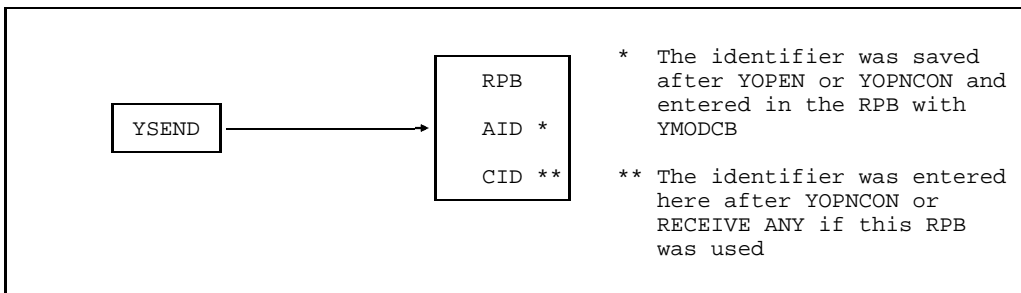
The RPB can also be modified with any macro call that addresses it (implicit RPB modification). It is up to the user either to create several RPB control blocks or always to address the same one and update its contents.

The application characteristics described in the ACB are transferred to DCAM with the YOPEN call. After YOPEN execution, DCAM returns an **AID (application identifier)** in the ACB. From then on all calls referring to this application may use the address of the ACB control block containing this identifier.



Example of control block addressing without identifiers

It is also possible to save the AID and to enter it in the RPB used for further calls.



Example of addressing an RPB in which valid identifiers were entered

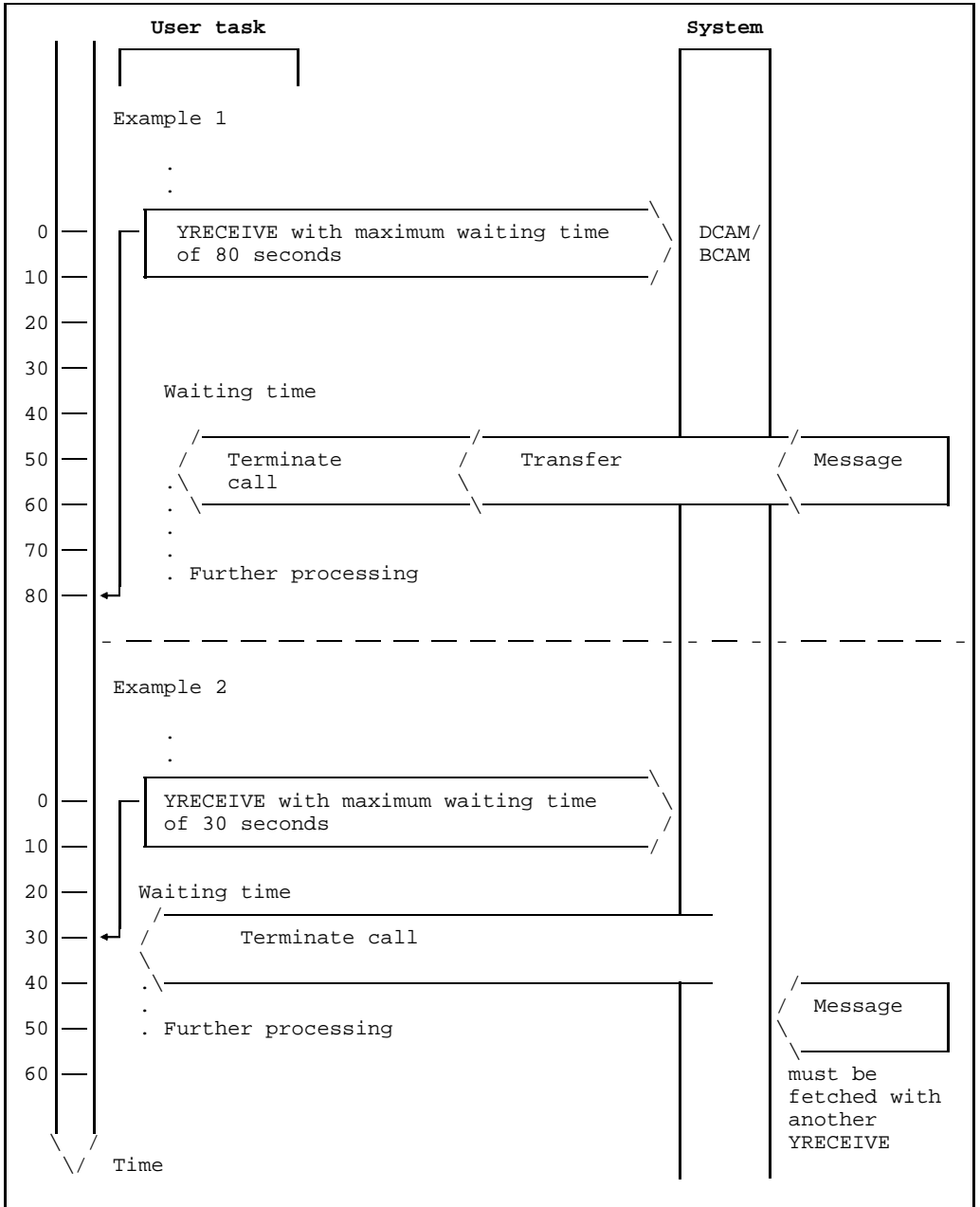
In this case the memory area for the ACB can be used for other purposes. The same applies to the description of the connection in the CCB and, in the case of DCAM(NEA) transport service applications, also in the DIP and the DCG. The **CID (connection identifier)** returned in the CCB and RPB can be used after an YOPNCON, for example. YSEND and YRECEIVE or YSENDREC then only require RPB control blocks containing the relevant AID and CID identifiers. If the entry was not made by DCAM, it can be entered with YMODCB provided the identifier was previously saved with YSHOWCB.

The use of the identifiers is advantageous because DCAM finds all the information required for the call in the RPB. This speeds up processing.

5.1.2 Synchronous execution of DCAM calls

In the case of synchronous execution of DCAM calls, control is only returned to the program by task management when this call has been processed by DCAM. If DCAM can process the call immediately there are no delays. However, with some functions, delays may occur due to DCAM having to wait for a response from the communication partner. When a connection is being set up (YOPNCON), the communication partner must contribute to this operation and when messages are to be received (YRECEIVE), the partner must have sent them and they must be available in the data area of the communication system and have been entered in a queue.

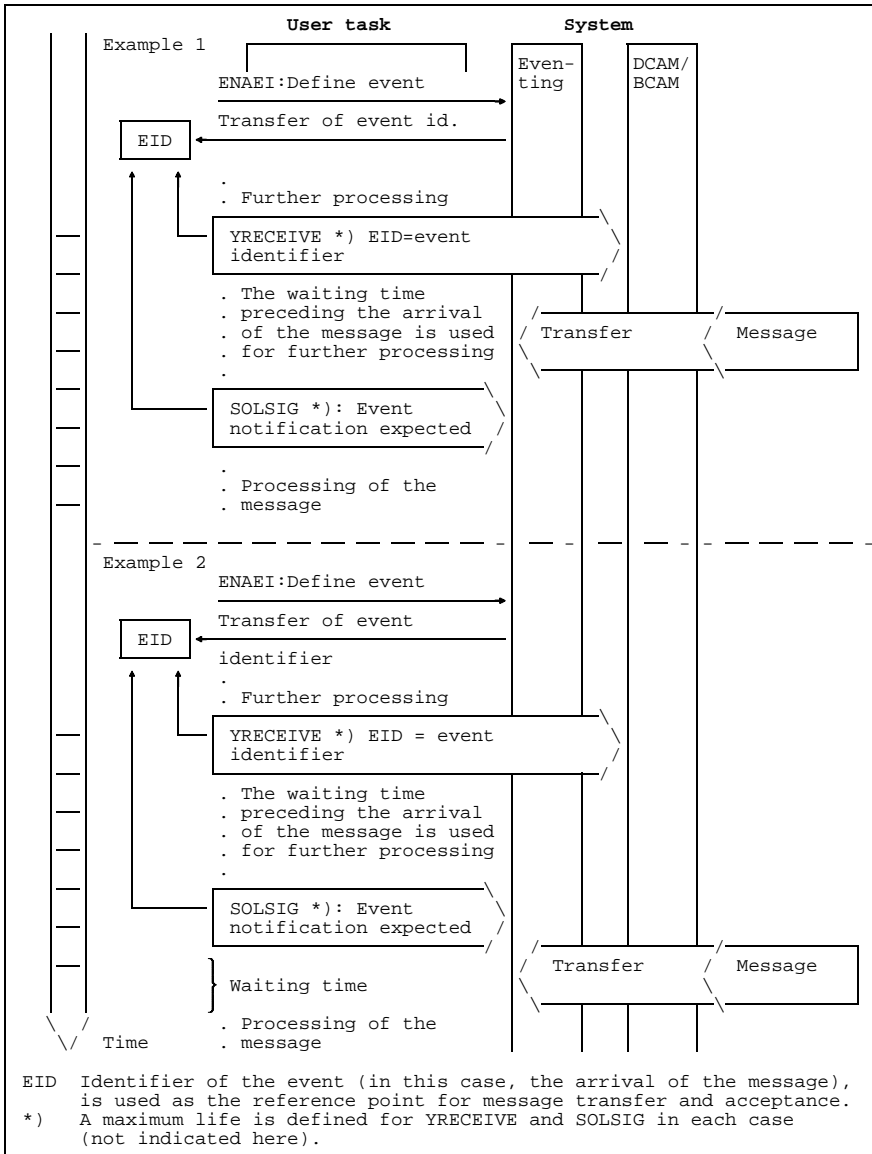
The programmer can decide whether and how long he is prepared to wait until the call can be executed. He defines the maximum waiting time for each call. However, he can also specify that a call should be terminated immediately even if the instruction cannot be performed. The call must then be repeated later in the program if required.



Examples of synchronous execution of YRECEIVE

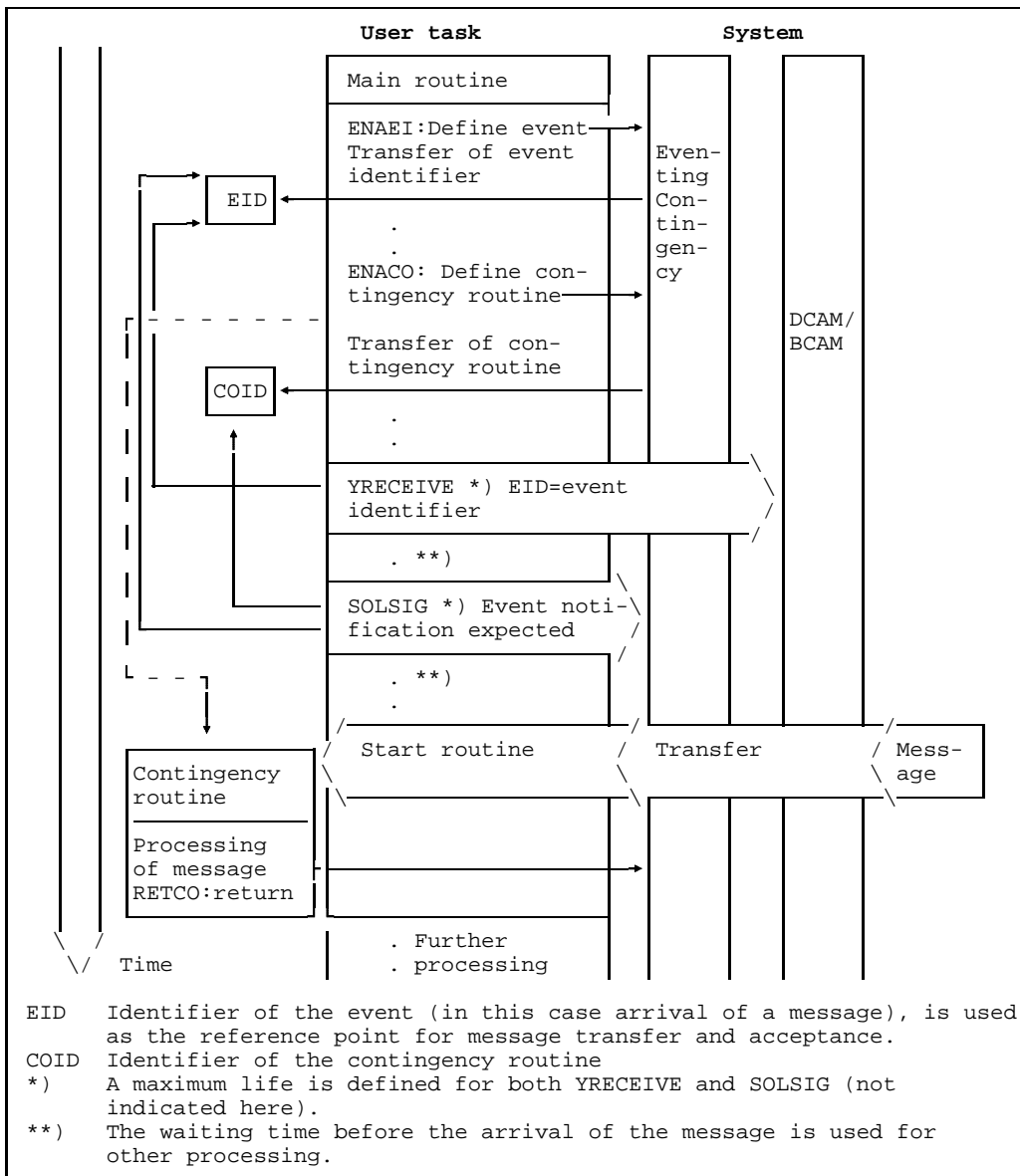
5.1.3 Asynchronous execution of DCAM calls

In the case of asynchronous execution of calls, control is returned to the program immediately when DCAM has accepted the call. This means that any delays can be used for other processing until the call is executed. For the purpose of asynchronous execution of calls, the user defines an event (ENAEI macro call) and specifies its identifier EID in the YOPNCON, YRECEIVE or YSENDREC macro. An event may be, for example, message transfer. The user can query this event (SOLSIG macro) at the location in the program at which he or she wishes to process the message, for example. The program can wait at this point if the message has not yet arrived.



Examples of asynchronous execution of YRECEIVE

However the user can also link the event with a contingency routine; this routine must be defined and its COID identifier must be available (ENACO macro call). In this case the user should query the event as soon as the DCAM call has been issued. When this event occurs the routine is started automatically.



Example with a contingency routine

It should be noted that a maximum lifetime has to be defined for the individual calls and that the lifetimes are related. The lifetime of the SOLSIG should be longer than that of the DCAM call, since the SOLSIG might otherwise have to be repeated.

Care should be taken to prevent actions in the program that assume completion of asynchronous processing without this actually having taken place (closing of the connection, for instance, while a YRECEIVE is being processed asynchronously).

Restriction:

Up to 8 instructions of the same type can be processed asynchronously at a given time (see page 138).

In order to be able to access values which were valid when the call was issued or to transfer the address of the control block used, the user can define event information.

This takes place when YOPNCON, YRECEIVE or YSENDREC is issued. It is entered when the event has arrived, either in a defined field (no contingency routine defined) or in a register (contingency routine defined).

5.1.4 Feedback information

The user receives feedback information after the termination of a DCAM call. This information consists of a 4-byte code which is entered in a field of the ACB or RPB control block, the FDBK field, and in register 15.

FDBK/Register 15			
Byte 1 (leftmost)	Byte 2	Byte 3	Byte 4
FDB1: Feedback summary	FDB2: Reason for rejec- tion (error code)	FDB3: Indicators	FDB4: Data indicators

The feedback information is made up of several bytes. By evaluating byte 1 (FDB1) the user will have sufficient information to estimate roughly what needs to be done. The user can either evaluate byte 2 (FDB2) or make do with a listing that permits later evaluation. The job on hand will ultimately be the decisive factor in this matter.

Bytes 3 and 4 (FDB 3-4) contain additional information, e.g. last data unit of a message received.

The feedback information is always in the FDBK fields of the appropriate control blocks.

In the case of a **synchronous call**, the feedback information includes information on the processing or rejection of the call.

In the case of an **asynchronous call**, only information on instruction acceptance or rejection is available in register 15 after completion of the call. Register 15 cannot contain any values relating to the execution of the instruction at this point. This is only available in the RPB block after the instruction has been executed. The FDBK field of the RPB control block can contain values relating to execution immediately after the asynchronous call is issued, e.g. if a message was already in the input queue when the asynchronous YRECEIVE call was issued or it contains information on acceptance or rejection like register 15.

The user can locate the RPB in the case of an asynchronous call by entering the address of the RPB in the EIDREF field. DCAM overwrites the first byte of EIDREF with X'0C'. In 31-bit mode, therefore, EIDREF2 should be used for addresses. The RPOSTAD field must then be 8 bytes long and "RPOSTL=2" must be specified for SOLSIG. After a SOLSIG call the user receives the RPB address via the RPOSTAD field or register 3 (contingency). Then he can access the RPB and with YSHOWCB he has access to the feedback information in the FDBK field. As the asynchronous call can be executed immediately after the DCAM call, only register 15 should be evaluated to check on acceptance or rejection.

The feedback information from the SOLSIG call (register 15) or the contingency routine (register 2) informs the user whether the event which occurred was the expected one or whether an error or a timeout occurred, but information on the execution of the call by DCAM is not provided (see SOLSIG in the BS2000 manual "Executive Macros").

In the case of YOPNCON and YRECEIVE, further information apart from the feedback information is provided, but only if the feedback does not indicate any errors i.e. FDB1=X'00' (see the relevant sections on the functions).

5.1.5 Asynchronous DCAM messages

Asynchronous notifications can be issued to the task by DCAM for a number of events in the data communication system which may occur in asynchronous relationship to program processing and which may have a decisive effect on processing (see table below).

Message	Transmitted by DCAM	Related to
LOGON	only to the primary task	connection function
PROCON		
SECOND		existence function
LOSCON		connection function
COMEND	to primary or secondary task	existence function
EXPR *		data transmission function
TACK *		

* for DCAM(NEA) transport service applications only

These events are:

LOGON

A connection request is received.

LOSCON

A connection was closed by the communication partner, the operator or due to an error.

A connection is about to be closed; warning (see /BCON and /BCTIMES commands in "Generating a Data Communication System").

Note

If the LOSCON event occurs without a warning, the connection is already cleared down; if the user now enters the macro call YCLSCON, it will be rejected with a return code $\neq 0$.

PROCON:

Partners already defined at communication access method generation time (XSTAT macro in "Generating a Data Communication System") were proposed for connection setup in the YOPEN or in a /BCIN operator command.

SECOND

A secondary task was opened or closed.



Applies to DCAM(NEA) transport service applications only:

The distribution code of a message is assigned to a distribution code name which is not opened by any secondary task. The primary task must first issue YFORBID for the distribution code name before it can receive the message proper.

COMEND

The communication access method was terminated.

Termination is pending (warning) (see /BCTIMES command in "Generating a Data Communication System").

The DCAM application has been terminated; message to secondary task.

The DCAM application will be terminated shortly; warning to connected secondary task (see /BCTIMES command in "Generating a Data Communication System").



EXPR An express message has arrived.

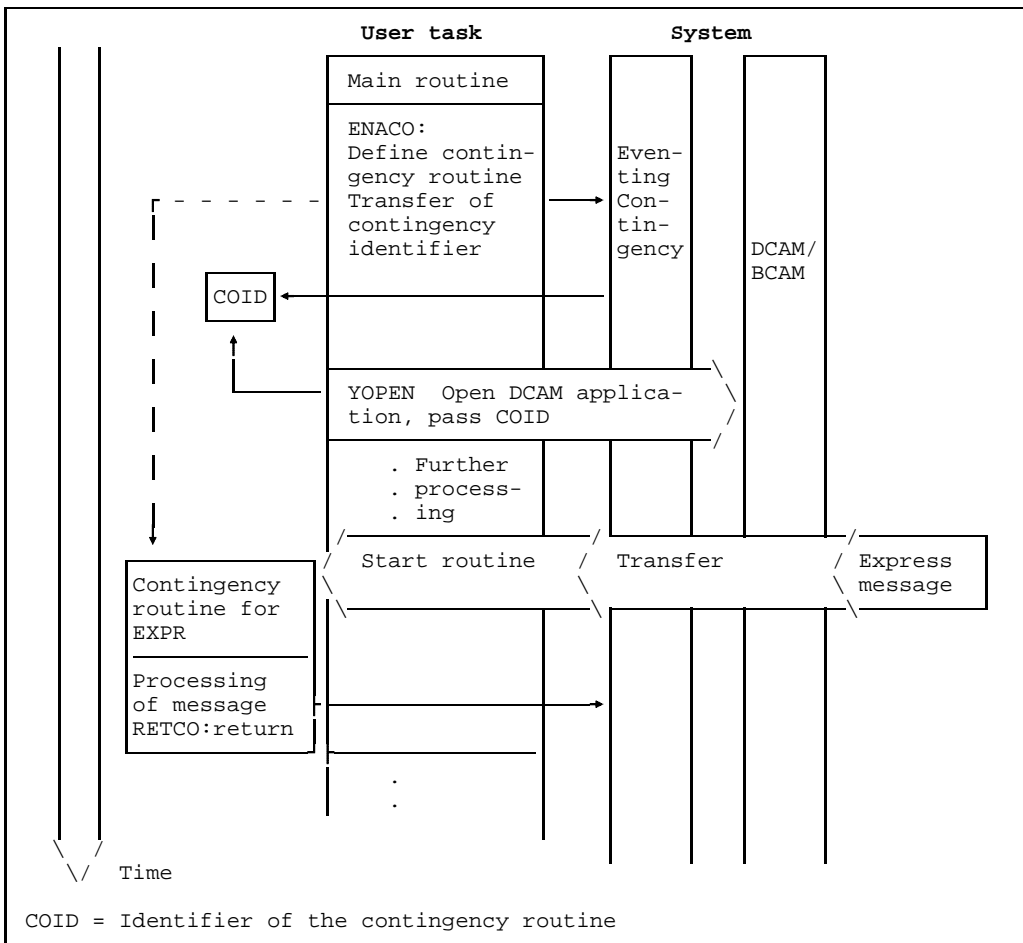


TACK A transport acknowledgment has arrived.

The DCAM application can be notified of the event concerning it if this is specified when the application is generated/opened. As the event notification results in a contingency routine being executed, a **contingency routine** must be **defined** for a notification to be received. The **ENACO macro** can be used for this purpose (see "DCAM Macros" manual). The returned identifier (COID) has to be transferred to DCAM via the ENB event control block (see page 50). The notifications which are to be accepted and the contingency routine which is to be initiated have to be defined at YOPEN time for the duration of an application.

Primary tasks are notified of all events, secondary tasks only of LOSCON, COMEND and, in the case of DCAM(NEA) transport service applications, EXPR and TACK. If a contingency routine is initiated, registers 1 to 8 contain all the information required for the processing of the event. The other registers have no defined contents. Input to the base register contents is the user's responsibility. Access to register contents of the interrupted routine or of the main routine is possible with the CONTXT macro. The priorities of the contingency routines are established when they are defined (ENACO) and modified via the LEVCO macro.

DCAM expects a response to each LOGON notification. The response does not have to occur within the contingency routine itself, just within a certain period of time defined during the generation of the communication system (/BCTIMES command). If no response is received within the time defined, this is interpreted as a rejection of the notification.



Example of express messages being received via an asynchronous DCAM notification (EXPR), for NEA only

The decision **not to accept** certain **notifications** while a DCAM application is being generated or opened has the following consequences:

- **LOGON** is not defined: Connection requests issued by the communication partners can only be processed by means of a YOPNCON call issued during the program run in case it is needed.
- **LOSCON** is not defined: Notification of connection loss can be obtained at the earliest from the feedback information from a call which refers to this connection.
- **SECOND** is not defined: The primary task must use other means (e.g. eventing) to discover that a secondary task has opened the application and which distribution code name it has.
 - ❗ Applies to DCAM(NEA) transport service applications only:
The primary task is unaware of messages assigned a distribution code name without a connected secondary task. These messages are not received by any task and are deleted after expiry of a system monitoring period.
- **PROCON** is not defined: Pre-defined proposals of connection setup are not reported to the DCAM application.
- **COMEND** is not defined: The fact that the DCAM application or the communication access method no longer exists is indicated in the feedback information at the earliest when the next call is issued to DCAM.
- ❗ **EXPR** is not defined: Express messages must be fetched with YRECEIVE in the order of the incoming messages.
- ❗ **TACK** is not defined: Transport acknowledgments must be fetched with YRECEIVE in the order of the incoming messages.

5.1.6 Reentrant ASSEMBLER programs

In BS2000, a program can control several tasks, i.e. the program is loaded only once for this number of tasks. The results are as follows:

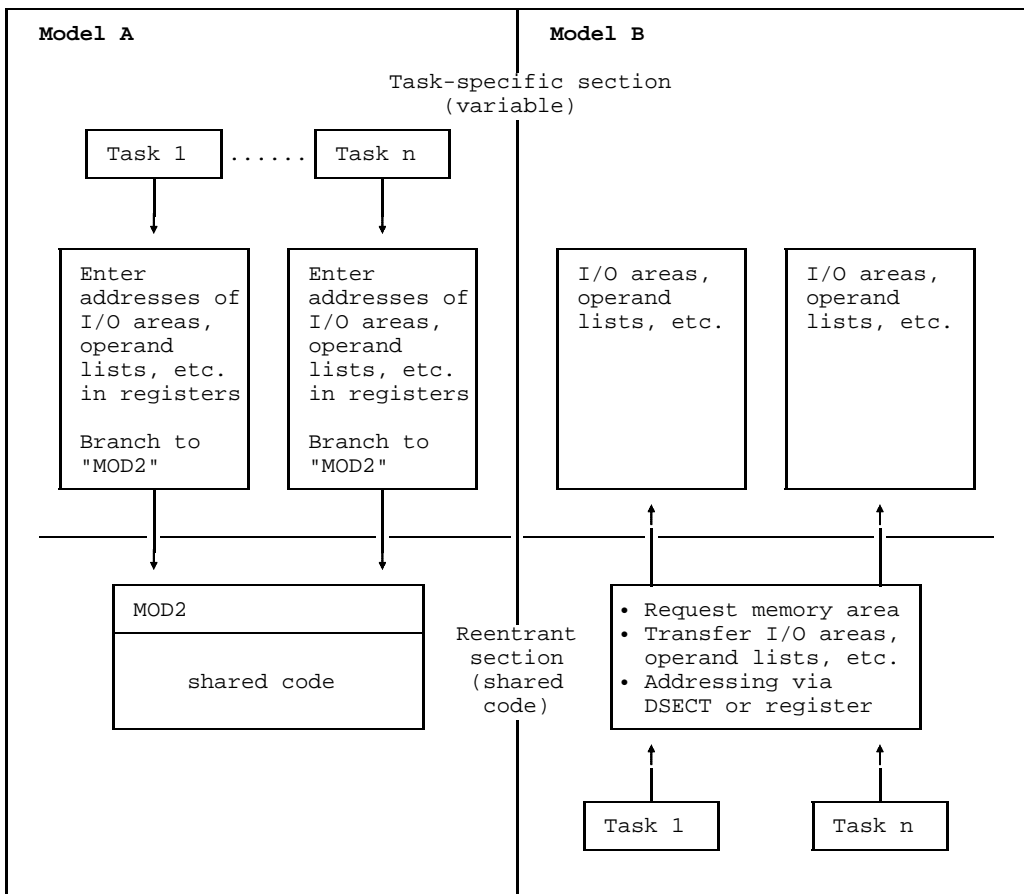
- **Loading times are shortened** as the program is loaded when it is called by the first task. When other tasks call it, the program is not loaded again, but used by them also i.e. it is shared.
- **Dialog response times are shortened** because the DCAM application, acting as the communication partner, has shared out the "burden" of incoming inquiries among several tasks. Timesharing task control is implemented although only one program is used (see page 27).
- The **system utilization is improved** because the paging rate is lower and the program is managed only once for several tasks in the main memory and the page memory.

This processing is only possible if such program modules, which are managed as '**shared code**' in the system, are invariable, i.e. reentrant. Since, however, task-specific I/O areas etc. are usually also required, this means that a program must be divided into a 'read-only' section and a variable section. This procedure is supported by DCAM as follows:

- **Control blocks** can be generated during the program run in a taskoriented area managed by DCAM or in a user area.
- By means of the **MF parameter** the parameter list and the operation code (SVC) of the macro can be separated for calls generating and handling control blocks.
- **Register notation** is used to a large extent.

The table below shows the possibilities available through reentrant programming and the required subdivision of the program.

Program section	Invariable (shared code)	Variable (task-specific)
Instructions	<ul style="list-style-type: none"> - Invariable instruction code using EX instruction, if necessary - Addressing of the variable section via registers (MF=E) - If required, provision of memory for the variable section (model B in figure below) 	<ul style="list-style-type: none"> - Instruction code including input to registers for branch to invariable module (model A in figure below)
Data and areas	<ul style="list-style-type: none"> - Number constants - Text constants - Address definitions (DSECT) - Operand lists (MF=L) 	<ul style="list-style-type: none"> - I/O areas - Computation fields - Save areas (registers) - Operand lists (MF=L)



Model for shared code programming

5.2 COBOL programs

A series of access modules to DCAM functions are provided in COBOL programs permitting synchronous and asynchronous processing. COBOL programs can control primary tasks but are also suited to controlling secondary tasks especially in the case of more complicated problems (see also page 27).

5.2.1 COBOL calls and data structures

All calls to DCAM are formulated as branches to subroutines (CALL...); the associated data structures with the parameters and data fields must be declared in the WORKING-STORAGE SECTION.

The following data structures are provided for:

– **Application structure (A-Struktur):**

The application structure contains the description of the DCAM application. This must be present at least once in the program.

– **Connection structure (V-Struktur):**

The connection structure contains the description of an instruction. This can exist for each connection or just once for several instructions.

– **Instruction structure (B-Struktur):**

The instruction structure contains the description of an instruction. This can exist for each instruction or just once for several instructions.

– **Wait structure (W-Struktur):**

The wait structure contains the description of the operands waiting for termination of asynchronous CALLs.



• **VTSU control block (VTSUCB-Struktur)**

The VTSUCB structure contains the VTSU parameters for input and output (see the VTSU User Guide).



• **Distribution structure (VTLG-Struktur):**

The distribution structure contains the description of message distribution with distribution codes. This is only required if message reception is to be controlled in this way.



- **Data structure (FHS-Struktur):**

This is only necessary if the data are to be formatted with the integrated FHS interface. (This requires the software product FHS version 3.0 or higher). The FHS modules must be available in the user TASKLIB.

The following COPY elements are available for all data structures named:

YDDCUAPL	for the A structure
YDDCUCON	for the V structure
YDDCUCOM	for the B structure
YDDCUWAI	for the W structure



The following are also available for DCAM(NEA) transport service applications:

VTSUCBC	for the VTSUCB structure
YDDCUDIS	for the VTLG structure
FHSMAINP	for the FHS operands

The following table shows which data structures are used in the calls of the individual subroutines. For some calls an additional area for data acceptance and transfer is required.

Some calls require still further areas which are not shown here (refer to "DCAM COBOL Calls").

Function	Subroutine	Data structures					VTSUCB	Area	FHS parameter 1)
		A	V	B	W	VTLG 1)			
Existence function Conne- ction function	YOPEN	x							
	YCLOSE	x							
	YOPNCON 2)	x	x	x		[x]			
	YCLSCON 2)	x	x						
	YCHANGE 2)	x	x						
	YREJLOG YSETLOG 1,2)	x	x		x		x		
Data communi- cation function	YSEND	x	x	x			[x]	x	[x]
	YRECEIVE	x	x	x			[x]	x	[x]
	YRSET	x	x	x					
	YPERMIT 1,2)	x				[x]		x	
	YFORBID 1,2)	x						x	
Wait function	YWAIT	x	x	x	x				

1) applies to DCAM(NEA) transport service applications only

2) only callable with primary task

The subroutines perform the following functions:

YOPEN	open a DCAM application
YCLOSE	close DCAM application
YOPNCON	set up a connection
YCLSCON	close a connection
YCHANGE	change connection characteristics
YSEND	send message
YRECEIVE	receive message
YREJLOG	reject connection setup request
YRESET	reset pending YRECEIVE calls
YWAIT	wait for DCAM event



The following are additional functions for DCAM(NEA) transport service applications:

YSETLOG	change status of a DCAM application
YPERMIT	allow receipt of data via distribution code
YFORBID	prohibit receipt of data via distribution code

A special convention applies to **status check** when this concerns the existence or connection function. **YINQUIRE** is used to call for system interrogation. The following inquiries can be made:

- **Which partner** wishes to establish a connection next, possibly with connection message receipt in LGMSG (TOP)?
- **How many partners** wish to establish a connection or have done so already (CNT)?
- **What is the status** of a certain DCAM application? (APP)



The following inquiries are also made for DCAM(NEA) transport service applications:

- **What are the partner characteristics** (PTN)?
- **Basic information on terminal** (BTI).
- **Description of terminal and character sets** (MCS).
- **Description of peripherals** (POT).

The A and B structures are used, and an area must be specified. The layout of this area varies depending on the function of the call. The function is defined in a special field.

Function	Subroutine	Function of YINQUIRE	Data structures		Areas 1)	LGMSG
			A	V		
Status inquiry for existence and connection function	YINQUIRE	TOP	X	X	X	[X]
		CNT	X	X	X	
		APP	X	X	X	
		PTN 2)	X	X	X	
		BTI 2)	X	X	X	
		MCS 2)	X	X	X	
		POT 2)	X	X	X	

1) Area depends on function

2) Applies to DCAM(NEA) transport service applications only

Note

The subroutines may also be called using the first four letters in a name, e.g. YOPN instead of YOPNCON.

To simplify programming, definition of the data structures once in a **source code file (PLAM)** is recommended. PLAM libraries can be generated and administered by the LMS utility routine. By means of the COPY statement, the stored data structures can be transferred to the program and, if necessary, modified.

5.2.2 Execute CALLs

5.2.2.1 Synchronous execution

In synchronous execution the next instruction after the branch to the subroutine is only executed once the DCAM call has been processed. For calls in which delays are likely, a maximum wait period can be defined. Delays may occur in the case of YOPNCON, for example: the wait for a connection setup request or acceptance of the connection request by the partner, or YRECEIVE: arrival of the message from the data communication system. The call is terminated after the specified period. If there is to be no wait, a "tentative" call is issued which must be repeated if necessary.

5.2.2.2 Asynchronous execution

Connection setup and the reception of messages may lead to delays. In order to increase data throughput, particularly when a large number of partners need to be served, the delays arising may be utilized for additional processing. The YWAIT enables the user to wait for some event. Once this event has occurred, he can resume processing.

Possible events:

- OPENED the YOPNCON request has terminated
- LETTER the YRECEIVE request has terminated
- GOSIGNAL the memory bottleneck has cleared
- LOSCON the connection was cleared by the partner or the system
- NOEVENT no DCAM event occurred during the wait period

5.2.3 Feedback information

After the return from a subroutine, i.e. after execution of a DCAM call, feedback information is stored in the data structure used (application structure or instruction structure).

The feedback information is subdivided into 3 fields:

- **return code**
- **error code**
- **indicator**

The return code provides a summary of the information encoded in the error code and the indicator. It will be needed, for instance, in order to branch to an error routine.

The indicator contains additional information after the execution of YRECEIVE.

Furthermore, the following entries are made in the instruction structure **after YRECEIVE**:

- the **actual length** of the message received even if the input area was smaller



The following additional entries apply in the case of DCAM(NEA) transport service applications:

- the **sequence number of the message received**
- the **sequence number of the acknowledged message** if a transport acknowledgment was received.

Additional information is passed in the event of extra calls (YOPEN, YOPNCON, YWAIT) - provided the acknowledgment does not indicate any errors (see section on 'Using the functions of DCAM' in the functional description).

5.2.4 Reentrant COBOL programs

The essential condition for the reentrant quality of a code is that it is invariable, i.e. the code is not modified in the course of execution (see page 115). Only then can it be managed and used in the system as shared code.

In employing overlay techniques, the COBOL compiler creates a root segment that is variable (non-reentrant) and various overlays that are, subject to a few restrictions, reentrant. The root segment contains the V constants for linking both the overlays as well as the COBOL runtime system (ITC...) and the CALL modules. This segment is assigned the name given in the PROGRAM-ID.

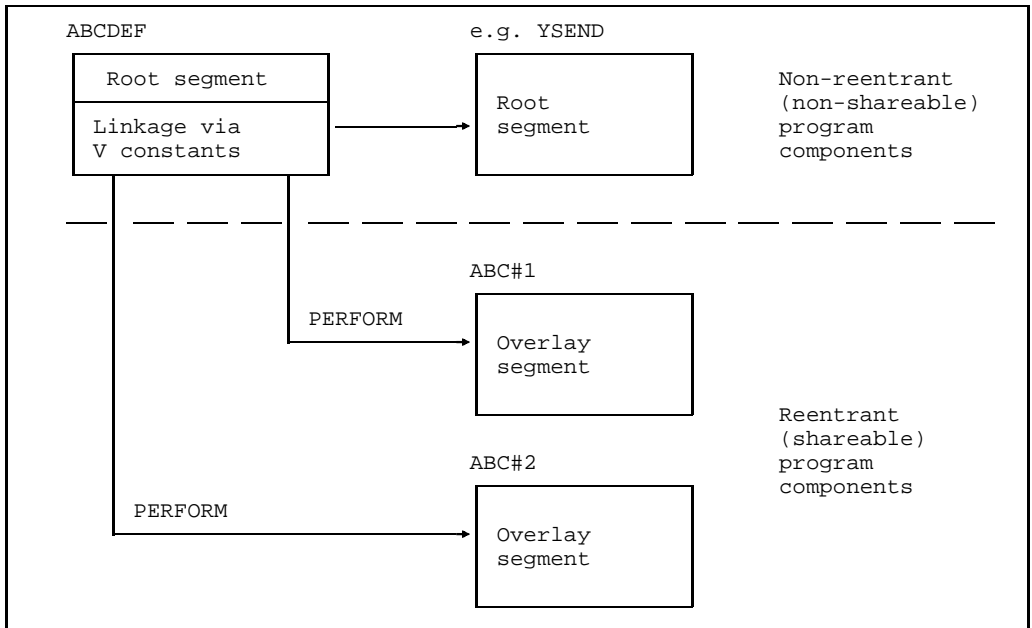
Note

If the COBOL85 compiler is used, the COBOL runtime system is also reentrant.

The overlays are called by the root segment via PERFORM. Their names are made up of the first three characters of the name given in the PROGRAM-ID, one special character (#) and the overlay number (e.g. ABC#50).

In these overlays, no DCAM COBOL modules may be invoked with CALL as these are variable (non-reentrant).

When the COBOL85 compiler is used, reentrant code is generated in the overlay segments.



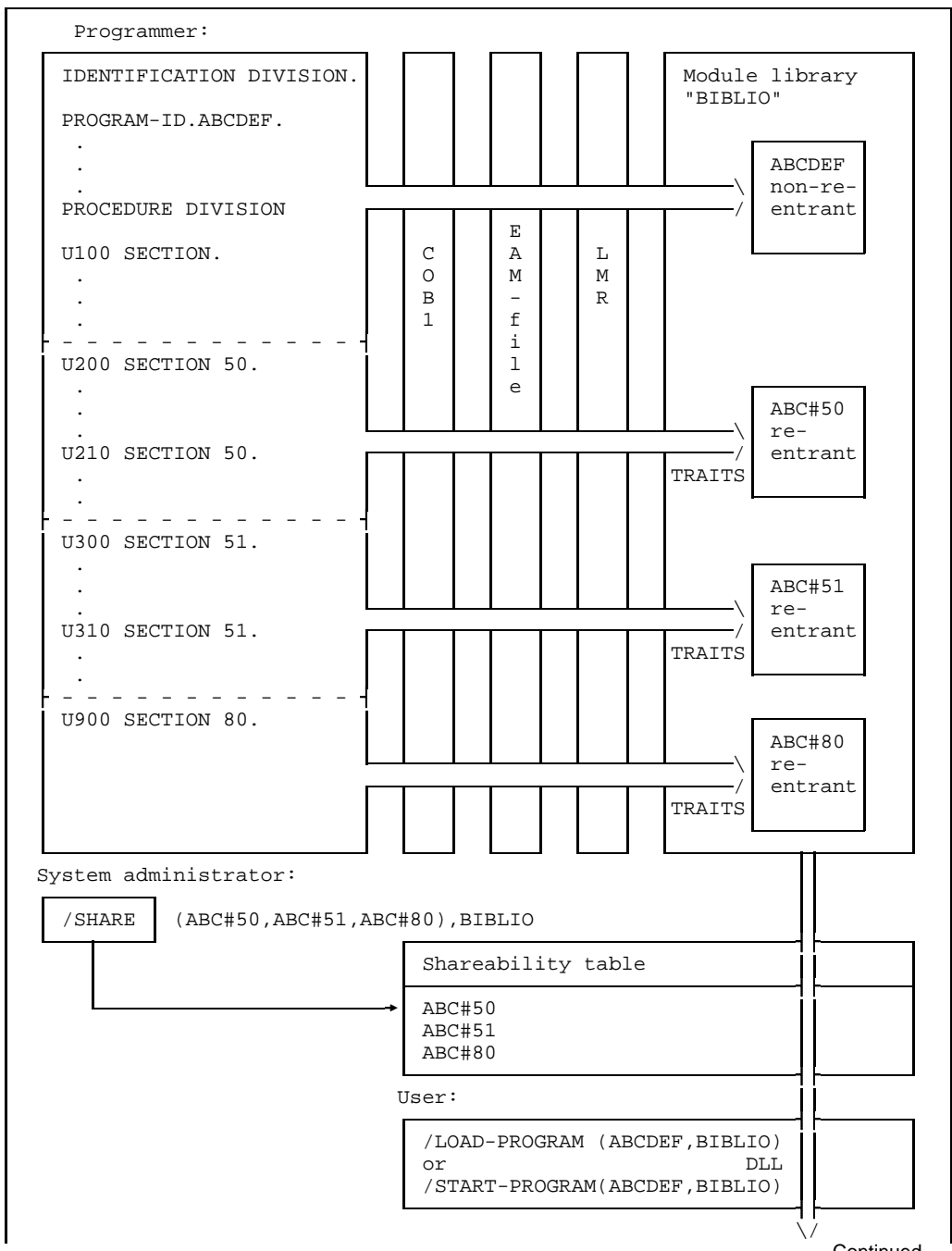
Segmentation of a COBOL program

In order to check overlay segments for reentrancy (read-only attribute), the TRAITS statement must be employed when they are entered in a module library by utility routine LMR. This is necessary because the COBOL compiler does not explicitly test this property at the present time.

For identification purposes, the ANSICOB compiler creates a linkage editor control card (OVERLAY...) at the beginning of a module. During the LMR run these cards are removed with a corresponding notification. This is necessary, and therefore appropriate, for further processing by the Dynamic Linking Loader (DLL). Only the Dynamic Linking Loader (DLL) is capable of loading shared code and must in this case be used. It is called with

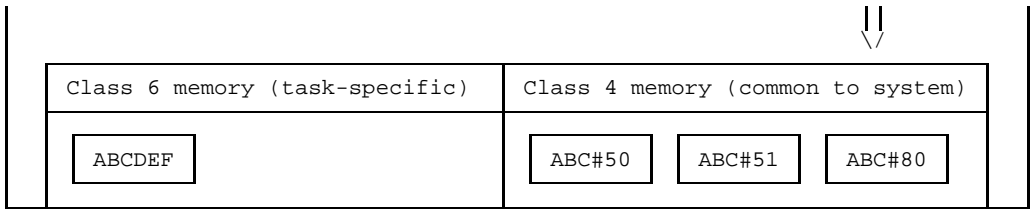
```
/EXEC (modulename, libraryname) or
/LOAD (modulename, libraryname)
```

Note, however, that shared modules are only loaded into the system's class 4 memory, where they are available to all tasks, if the system administrator has entered them into the appropriate system module tables during initiation of the BS2000 session (cf. "System Controller's Guide" SHARE command).



Continued -

Continued



Generation of shared code

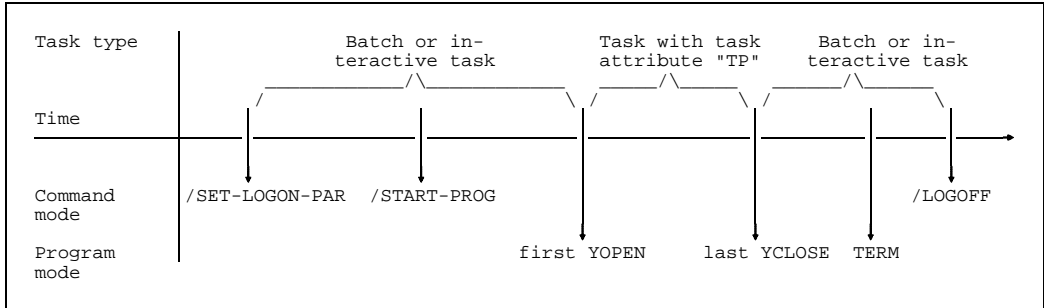
5.2.5 Using other system interfaces

There are no limitations on a DCAM COBOL program within the framework of COBOL use in BS2000.

Note the possibility of using the SHARED UPDATE mode for the USER, PAM and ISAM access methods of the data management system. This mode provides the necessary protection mechanisms for access of a task group to a file.

5.3 Execution of a DCAM program: DCAM task

At first a DCAM task is just any BS2000 task. As soon as a program opens a DCAM application within this task, the task attribute is set to "TP" (transaction processing) provided the JOIN allows this.



Task types

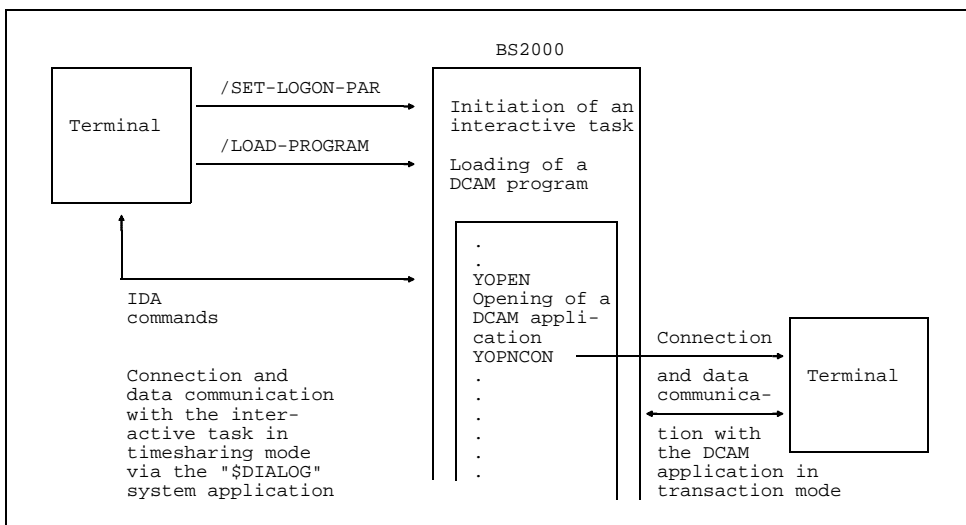
5.3.1 Starting a DCAM task

Before a task can become a DCAM task, it is started as a batch or interactive task with the /SET-LOGON-PARAMETERS command. This command can be issued

- for **batch tasks** from other tasks by means of **ENTER files** which contain a /SET-LOGON-PARAMETERS command at the beginning. Batch tasks are started by the system if the resources are available and the limitation defined by the system administrator allows it.
- for **interactive tasks** from any **interactive terminal**, e.g. an 9750 Data Display Terminal. In this case, the task is started immediately.

Note

The connection between a terminal and an application is set up as early as during the pre-dialog. In the case of interactive tasks, this is the connection between the terminal and the "\$DIALOG" application. After such a task has become a DCAM task, the connection to the terminal remains in existence and it is thus not possible to set up a connection from this terminal to the DCAM program controlling the task. However, it is possible to check and modify the program from this terminal using IDA.



Timesharing and inquiry-and-transaction processing in a task

5.3.2 Terminating a DCAM task

A DCAM task is returned its original task attribute when the program controlling it closes the last DCAM application or is terminated. It is then managed as a batch or interactive task as before and is terminated with the /LOGOFF command. This command can be issued

- as the last command in an ENTER file
- as input from an interactive terminal
- in the program mode by means of the LOGOFF macro or indirectly by means of the TERMJ macro if a branch is made to /LOGOFF.

Tasks can also be terminated by means of

- the operator's /CANCEL-JOB command or
- the /CANCEL-JOB command in the interactive task for tasks with the same user ID, but different TSNs
- the operator's /SHUTDOWN command
- by abnormal task termination in the case of a system error.

5.3.3 Notes on programming

This section describes typical stumbling blocks in programming. You will find it of assistance in working with DCAM applications.

Connection setup does not work:

- EDIT=SYSTEM is not permitted for communication between applications (also applies to APS applications)
- EDIT=USER is not permitted for MSN terminal
- If YOPNCON OPTCD=ACQUIRE, edit parameters are merely proposals. The next YOPNCON with this YCCB evaluates the DEXP (data exchange protocol) of the last connection instead of the generated value.

Data-flow problems:

- a DCAM application is restricted to synchronous send. Synchronization with transport acknowledgments is inadequate for connections to printers. Print acknowledgments must be specified.

Message is lost:

- connection is in the wrong CS/CA state. Note that CS/CA is not evaluated when acknowledgments are received.

Negative acknowledgment (NACK) received, although no acknowledgment was requested:

- YSEND allows only for the reception or non-reception of positive acknowledgments. Negative transport acknowledgments can be suppressed only by specifying NOTACK in the YOPEN.
- with EDIT=SYSTEM messages can be divided only by VTSU (e.g. RESET messages). More than one acknowledgment can be issued per message.

Return code X'1808' after YOPNCON:

- the address pointing to the YCCB was destroyed by a previous YOPNCON. you must send YOPNCON with CCB=CCBADR.

Return code X'20000000':

- DCAM does not guarantee downward compatibility. Phases compiled with DCAM V8.0 are not executable in the DCM V7.01 environment, even when the YGENCB macro is used.

Line feed before every YSEND:

- in contrast to TIAM (WROUT) a user entry is possible at any time in DCAM. This is why a new line is started before every SEND.

Last message is lost:

- a DISCON can overtake normal messages. The acknowledgment for the last message must be received before the connection is cleared down.

DCAM clears down a connection:

- this may happen if an application fails to fetch a large number of transport acknowledgments.

Notes on performance:

- PRIM=TASK must not under any circumstances start a PASS or VPASS loop. Use SOLSIG instead
- A VPASS after YOPCON has been unnecessary since BS2000 V6.0
- When transferring files via MSV2 connections (e.g. teleprinter connections), it is always advisable to use ELEMENT.
- Connections with EDIT=USER require significantly less input of system resources than connections with EDIT=SYSTEM.

6 Appendix

6.1 DCAM calls

Types of DCAM call:

- Macro calls (ASSEMBLER = A)
- COBOL calls (COBOL = C)

Type	Call	Function	Description
A	YACB	1)	Generate application control block
A	YAPPL	Name assignment	Store data for DCAM application in CLT; delete this data
A	YCCB	1)	Generate a connection control block
A; C	YCHANGE	Connection	Change characteristics of a connection already established
A; C	YCLOSE	Existence	Close down a DCAM application
A; C	YCLSCON	Connection	Cancel a request or close down a connection
A	YCONN	Name assignment	Store connection data in CLT; delete this data
A	YDCG 2)	1)	Generate a distribution code group block
A	YDDACB	1)	Generate a (dummy) section for control block ACB
A	YDDCCB	1)	Generate a (dummy) section for control block CCB

Type	Call	Function	Description
A	YDDDCG 2)	1)	Generate a (dummy) section for control block DCG
A	YDDDIP 2)	1)	Generate a (dummy) section for control block DIP
A	YDDENB	1)	Generate a (dummy) section for control block ENB
A	YDDRPB	1)	Generate (dummy) section for control block RPB
A	YDIP 2)	1)	Generate a distribution parameter block
A	YENB	1)	Generate an event notification block for relating asynchronous messages to contingency routines
A; C	YFORBID 2)	Data communication	Cancel the assignment of a distribution name to a distribution code group
A	YMODCB	1)	Change items in existing control blocks
A; C	YOPEN	Existence	Open a DCAM application
A; C	YOPNCON	Connection	Set up a connection
A; C	YPERMIT 2)	Data communication	Assign a distribution name to a distribution code group
A; C	YRECEIVE	Data communication	Receive a message
A; C	YREJLOG	Connection	Reject a connection request
A; C	YRESET	Data communication	Delete receive calls; change connection state CS/CA

Type	Call	Function	Description
A	YRPB	1)	Generate a request parameter block
A; C	YSEND	Data communication	Send message
A	YSENDREC	Data communication	Send and receive message combined
A; C	YSETLOG 2)	Existence	Change status of an application
A	YSHOWCB	1)	Transfer items from a control block into the user area
A	YTESTCB	1)	Compare an item in a control block with a given value
C	YWAIT		Wait for termination of asynchronous CALLs
A	YGENCB	1)	Generate one or more control blocks of any kind
A; C	YINQUIRE	Existence; connection	Retrieve information on applications and connections

1) Control block functions

2) For DCAM(NEA) transport service applications only

6.2 Limit values

– Asynchronous processing

The following table shows how many asynchronous instructions may be processed concurrently.

Maximum number	Valid for call of type	Function
8 per application	YOPNCON ACCEPT ANY	Accept request from any partner
8 per application	YOPNCON ACCEPT SPEC	Accept request from a specific partner
128 per applic.	YOPNCON ACQUIRE	Issue request
8 per task of an application	YRECEIVE ANY	Receive a message from any partner
8 per connection	YRECEIVE SPEC	Receive a message from a specific partner

– Use of distribution codes: for NEA transport service only

The following table shows the maximum values for the definition and assignment of distribution codes.

Type	Upper limit	Applicable to
Number of distribution code groups per virtual connection	8 16	COBOL ASSEMBLER
Distribution codes per code group	8	ASSEMBLER COBOL
Tasks per distribution name	8	ASSEMBLER COBOL

– **A task**

can keep a specific **number of DCAM applications** open **concurrently**. The precise quantity is to be found in the relevant release notice for the BS2000 operating system.

The number of "non-predefined applications" is also limited. This value can relate to a task or to the system as a whole. These limiting values are defined at system start and can be modified during operation.

An application can maintain only a certain number of connections simultaneously. There is also a limit to the number of connections a "non-predefined application" can employ (see above).

– **Maximum message length (MAXLN)**, for NEA transport service only

Local parameter which influences the economy of the buffers provided by the system. It contains the maximum length of the data being sent (Transport Service Data Unit, TSDU).

When EDIT = USER:

1 message = 1 TSDU (YSEND)

When EDIT = SYSTEM, EDITOUT = PHYS or FORM:

DCAM transmits the system-edited message in segments, the size of which is determined by MAXLN and the device capacity. It is up to the user to ensure the device capacity is not exceeded.

Any records longer than MAXLN are truncated during editing.

Maximum length for a user message per YSEND: 32767 bytes

Note

An edited record is always longer than user data as control characters are converted and protocol labels added.

The following table shows the maximum values of MAXLN:

Requested MAXLN	←- 65530	Default value	
DCAMVER	8.0	8.0	Default value
with DVR MAXLN	x = 65530(*)	4096	4096
with DAST		4096	32767

(*) The results depend on the hardware/software configuration and generation.

– Limit values for resources

The following static maximum values apply in DCAM V11.0:

GID/application (permitted)	32
GID/application (not permitted)	32
DID/application	32

The number of applications and the number of P1 events is restricted by the BS2000 name manager to around 500 per task in the case of BS2000 V9.5 and to around 2000 in the case of BS2000 V10 and higher.

In BS2000 the number of P1 contingencies generated for a task but not yet executed is limited.

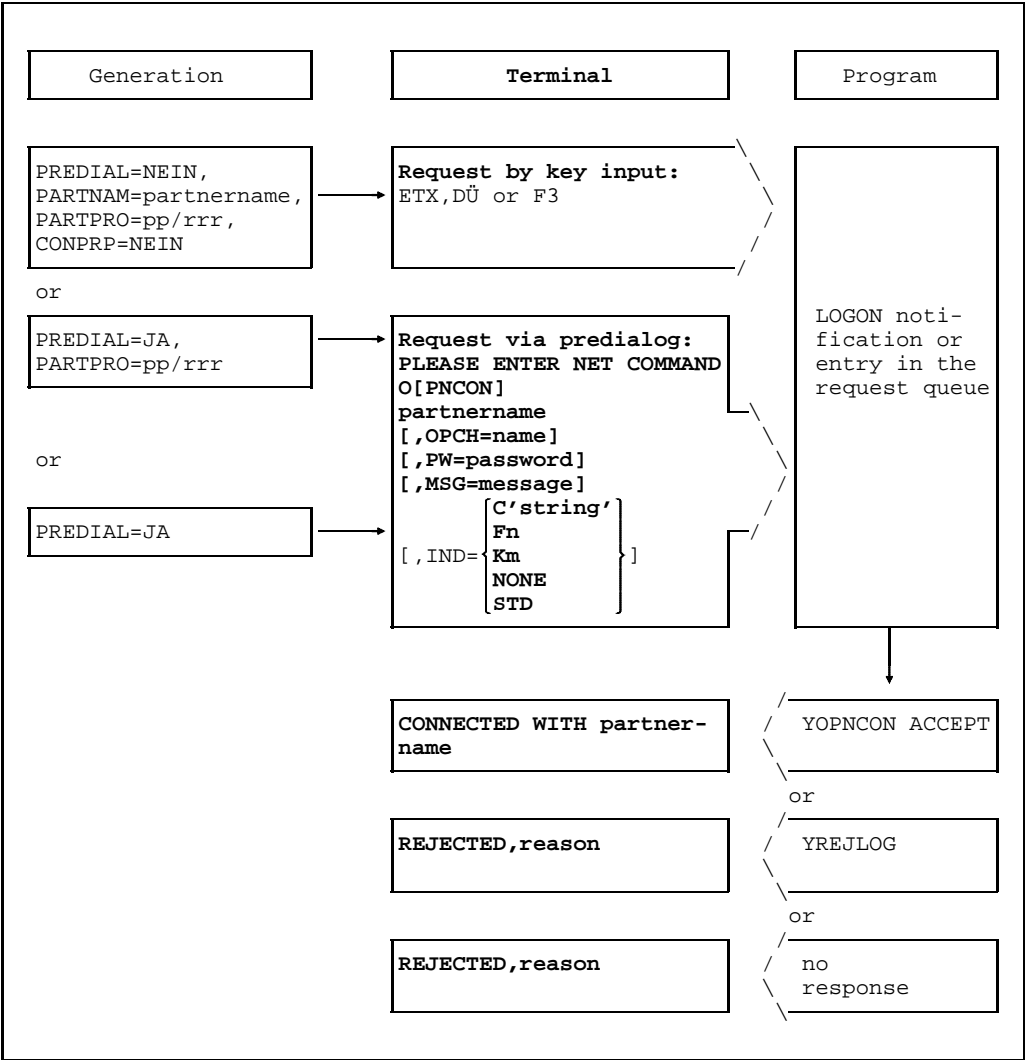
6.3 Setting up a connection from a terminal



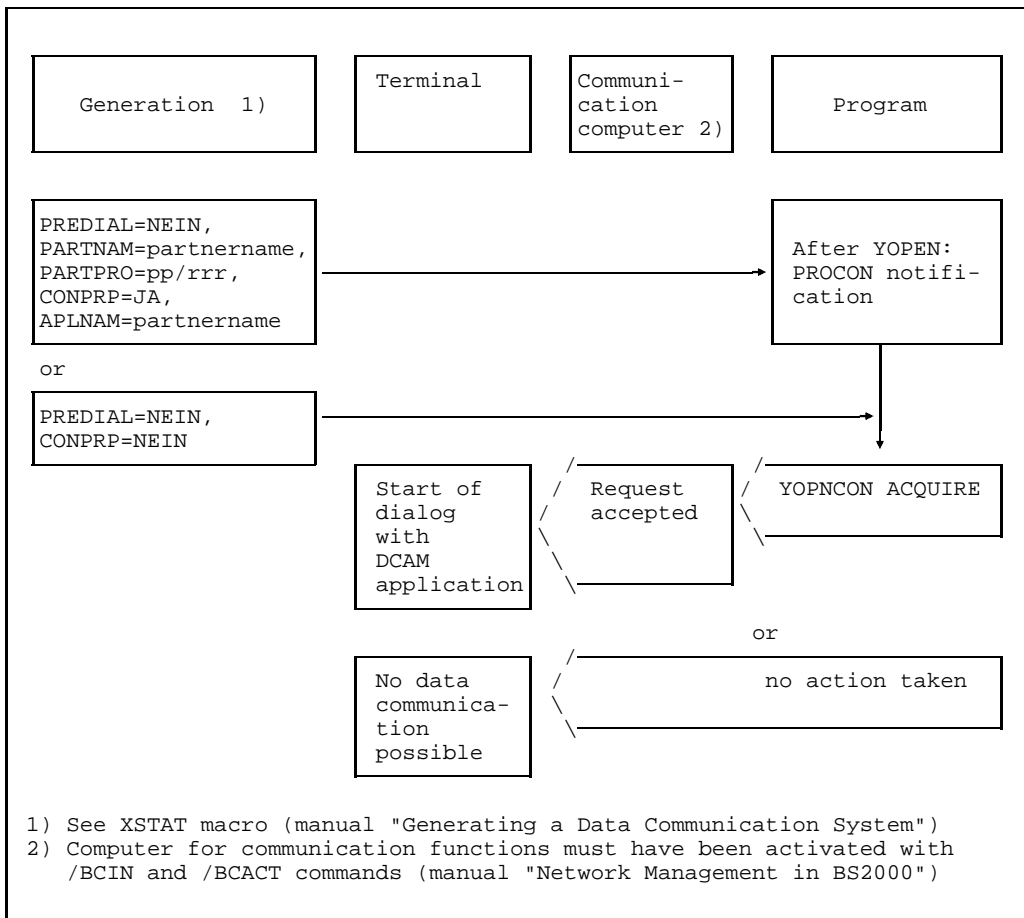
This section applies to DCAM(NEA) transport service applications only.

Setting up a virtual connection from a terminal is an undertaking which varies in accordance with the way the terminal itself was generated, with the options that have been preset and with the constraints imposed by the relevant DCAM application as the communication partner. The following diagrams will try to illustrate two fundamentally different approaches to this:

- the DCAM application required as the prospective partner has already been opened;
- the terminal has been turned on and is operational;
- the physical connection at the level of the transmission line has been or is being set up in one of the following three ways:
 - the person attending the terminal has set it up;
 - the physical connection is a dedicated line and requires no further setup;
 - the communication computer to which the terminal is connected sets up the line connection in response to a YOPNCON ACQUIRE call from the DCAM program.



Initiative lies with the DCAM application (LOGON attribute)



Initiative lies with DCAM application (NLOGON attribute)

Glossary

CMX application

A communication application running on a SINIX or BS2000 computer and controlled by a CMX application program.

communication application

A facility for processing the messages exchanged by communication partners. It is addressed by the data communication system via its access point.

communication partners

Entities that maintain connections and exchange data with each other.

[communication] protocol

A description of the conditions and formats for transfer of information between equivalent functional layers in the data communication system.

communication computer

A computer designed specially for communication functions.

communication access method

The software that provides applications with an interface to the communication facility.

connection

A relationship between two communication partners that permits them to exchange data.

data unit

The quantity of data that can be passed to or received from DCAM with one call.

data communication system

A complex combination of hardware and software products that permits communication partners to exchange data in accordance with certain rules.

[DCAM] application

A communication application that is controlled by at least one DCAM application program.

[DCAM] application program

A program that uses the services of the DCAM access method; it controls one or more DCAM applications.

[DCAM] data transmission function

A DCAM function that is related to the transmission and reception of messages and acknowledgments.

[DCAM] event

A DCAM-specific event that can be used for coordination of certain operations in the data communication system. There is no specific time relationship between its arrival and the execution of the program (= asynchronous event).

[DCAM] existence function

A DCAM function that is related to the generation and cancellation of DCAM applications.

[DCAM] name assignment function

A DCAM function that permits the user to generate application programs independently of static parameter values, such as the DCAM application name, the partner name, etc.

[DCAM] connection function

A DCAM function that is related to the establishment and clearing down of connections.

express message

A message, with a restricted length, that is transmitted with a higher priority than normal messages.

format terminal

An operating mode of a virtual terminal where the message consists of a format (= entry form, screen mask).

ICMX

General C interface offering the OSI transport functionality

line terminal

An operating mode of a virtual terminal where the message is structured in the form of lines.

logical terminal

--> virtual terminal

message

A logically related set of data that is to be transmitted to or received from a communication partner.

process

A facility for executing a program within a task.

shareable DCAM application

A DCAM application that can be used simultaneously by more than one task.

task

The carrier for processes. In BS2000, tasks are used, amongst other things, for execution of user jobs (e.g. batch job, interactive task) or for operation of (DCAM, UTM, TTX) applications (execution of all procedures specified between the BS2000 commands LOGON and LOGOFF).

terminal user

A person who uses a terminal to exchange data with a communication partner.

transport service

A service for the exchange of data between communication partners. The transport service initiates and monitors the transport of messages through the data communication system and manages connections.

transport acknowledgment

An event that provides information about the successful or unsuccessful execution of a data transfer.

virtual terminal (logical terminal)

A terminal model whose functions are mapped on the physical characteristics of various terminal types.

XHCS

Extended Host Code Support

Software supporting 8-bit terminals.

Related publications

DCAM (BS2000/OSD)

Macros

User Guide

DCAM (BS2000/OSD)

COBOL Calls

User Guide

CMX (BS2000)

Communication Method in BS2000

User Guide

COBOL85 (BS2000)

COBOL Compiler

User Guide

COBOL85 (BS2000)

COBOL Compiler

Reference Manual

Assembler (BS2000)

Reference Manual

Assembler Instructions (BS2000/OSD)

Reference Manual

ASSEMBH

User Guide

TIAM (BS2000/OSD)

User Guide

FHS (BS2000/OSD)

User Guide

VTSU (BS2000/OSD)

User Guide

XHCS

Extended Host Code Support for BS2000/OSD

User Guide

BS2000/OSD

Executive Macros

User Guide

BS2000/OSD

Utility Routines

User Guide

BS2000/OSD

Commands

Volume 1-7

User Guide

BS2000/OSD

System Exits

User Guide

SOCKETS-DE (BS2000)

Communication Method in BS2000

User Guide

Ways to Open Communications

The ISO-Reference Model in the Context of Communications

Brochure

Index

A

- A-Struktur 118
- ACB 97
- access protection 24
- access to terminal 37
- accompanying information 57
- acknowledgment, negative 71
- addressing 19
- ANY 53
- ANYSSTART 61
- application 17
 - alter state 48
 - close 49
 - non-shareable 42
 - open 29
 - query 49
 - shareable 42, 44
- application control block 97
- application structure 118
- APPSTART 61
- ASSEMBLER program 97
- assign distribution code name 90
- asynchronous execution of calls 105
- asynchronous notification 111

B

- B-Struktur 118
- batch task 129

C

- CA state 83
 - change 88
- CALL
 - asynchronous execution 123
 - synchronous execution 123

- CALLs 135
- calls 24
- cancel assignment 91
- cancel receive macro 88
- CCB 99
- change characteristics 75
- characteristics connection 55
- CLT 92
- COBOL calls 24
- COBOL program 118
- code length 69
- COMEND 112
- common receiver queue 20, 83
- communication link table 31
- communication partners 16
- connection 19
 - close 76
 - close down explicitly 29
 - close down implicitly 29
 - definition of the 57
 - establish 29
 - predefined 71
 - setup 50
 - terminate 29
- connection characteristics 55
- connection control block 99
- connection function 50
- connection setup 63
- connection structure 118
- contingency routine 51, 107
- control block 97
- control block generation
 - dynamic 102
 - static 101
- controlling task 32
- COPY elements 119
- COPY statement 122
- CS state 83
 - change 88

D

- data 15
- data flow control 26, 57
- data length, maximum 61
- data structure 119
- data transmission 77
- data unit 34
 - receive 83
 - transmit 78
- DCG 99
- definition of the connection 57
- DIP 99
- DISCO 46
- DISNAME 46, 47
- distribution code 46, 47
 - implicit 24
- distribution code assignment 88
- distribution code group 88
- distribution code group block 99
- distribution code name 46
- distribution code-oriented queue 22
- distribution codes 22
 - use of 68
- distribution parameter block 99
- distribution structure 118
- dynamic generation of control block 102
- dynamic name assignment 31

E

- EDIT 55
- EDIT options 61, 96
- ENACO 107, 112
- ENB 97
- entries, query 73
- error code 124
- error handling routine 31
- event 105, 111, 123
- event information 108
- event notification block 97
- execute program 129
- execution of calls
 - asynchronous 105
 - synchronous 103
- existence function 41

EXPR 112
express message 26, 83, 112

F

FCB 92
FDBK field 109
feedback information 31, 109, 124
FHS-Struktur 119
form terminal 95
format terminal 81

G

GO signal 26
GOSIGNAL 123

H

header length byte 81

I

implicit distribution code 24
indicator 124
instruction structure 118
interactive task 129
invariable 115
ISO attribute 42
ISO transport services 9

L

layer 9, 10
LETTER 123
limit values 138
line terminal 80, 96
logical line 80
LOGON 48, 111
logon password 61
LOSCON 111, 123
LOSCON routine 71

M

macro calls 24, 97, 135
maximum length of data 61
maximum message length 58, 63
maximum waiting time 103
MAXLN 61
message 34
 receive 29, 83
 transmit 29, 78

message code 60
message distribution with distribution codes 61
message editing 61, 80
message header 81
message length, maximum 58, 63
messages 24
MF parameter 115
more-data function 34, 58

N

name assignment, dynamic 31
name assignment function 92
NEA transport services 8
negative acknowledgment 71
NOEVENT 123
non-shareable application 42
notification, asynchronous 111
NSHARE 43

O

open application 42
open system 9
OPENED 123
opening
 primary 44, 46
 secondary 46
originator-oriented queue 20, 83
OSI 9
OSI Reference Model 9, 10
overlay 125

P

parallel connection 19, 55
parameter values 98
partner name 57
password 24
 logon 61
physical programming 37
PLAM library 122
predefined connection 71
primary opening 44, 46
primary task 17, 32
PROC 55
processor name 29, 57
PROCON 111

PROCON notification 54
program 15
programs, reentrant 115, 125

Q

queue
 common receiver 20, 83
 distribution code-oriented 22
 originator-oriented 20, 83

R

RDF password 25
reentrant 125
reentrant programs 115, 125
reference model 9
request
 accepting a 66
 delete 75
 reject 75
request parameter block 97
return code 124
RLTH 58
root segment 125
route selection 60
RPB 97

S

SECOND 112
secondary opening 46
secondary task 17, 33
SHARE 46, 47
shareable application 42, 44
shared code 115
SOLSIG 105
source library 122
SPEC 53
START state 48
static control block generation 101
status check 121
synchronous execution of calls 103

T

table, task-oriented 92

TACK 112

task 15, 129

 controlling 32

 start 129

 terminate 131

task-oriented table 92

terminal 17

 access to 37

 form 95

 format 81

 line 80, 96

 virtual 37, 95

terminal status 60

TFT 92

TIAMCTRC 96

TIDU 34

TIMEOUT 71

transport acknowledgment 76, 87, 112

TSDU 34

U

use of distribution codes 68

UTM application 39

V

V-Struktur 118

variable 115

virtual terminal 37, 95

VTCSET 96

VTLG-Struktur 118

VTSU control block 96, 118

VTSUCB-Struktur 118

W

W-Struktur 118

wait structure 118

waiting time, maximum 103

WORKING-STORAGE SECTION 118

X

X.25 interface 99

Y

YAPPL 93

YCHANGE 75

YCLOSE 49

YCLSCON 75, 76

YFORBID 91

YGENCB 102

YINQUIRE 49, 73, 121

YOPEN 42

YOPNCON 50

YPERMIT 90

YRECEIVE 83

YREJLOG 75

YRESET 88

YSEND 78

YSENDREC 88

YWAIT 123

Contents

1	Preface	1
1.1	Summary of contents	4
1.2	Changes since the last version of the manual	5
2	Introduction to the DCAM interface	7
2.1	The Data Communication Access Method DCAM	8
2.2	Traffic relations in the data communication system with DCAM	12
2.3	Basic concepts	15
2.3.1	Program, data, task	15
2.3.2	Communication partners	16
2.3.3	Addressing	19
2.3.4	Connections	19
2.4	Characteristic features of DCAM	20
2.4.1	Distribution of incoming messages	20
2.4.1.1	Originator-oriented queue and common receiver queue	20
2.4.1.2	Distribution code-oriented queue	22
2.4.1.3	Implicit distribution code	24
2.4.2	Calls and notifications	24
2.4.3	Protection against unauthorized access	24
2.4.4	Express messages	26
2.4.5	Data flow control	26
2.5	Program structure	27
2.5.1	Functions of a DCAM program	27
2.5.2	Basic structure of a DCAM program	29
2.5.3	Control of primary and secondary tasks	32
2.5.4	Messages and local data units - more-data function	34
2.5.5	Access to terminals	37
2.6	Implementation of distributed processing	38
2.6.1	A DCAM application as a partner	38
2.6.2	UTM application as a partner	39

3	DCAM functions	41
3.1	Existence function	41
3.1.1	Open a DCAM application	42
3.1.1.1	Non-shareable DCAM application	43
3.1.1.2	Primary opening of a shareable DCAM application	44
3.1.1.3	Primary opening - use of distribution codes	46
3.1.1.4	Secondary opening	46
3.1.1.5	Secondary opening - use of distribution codes	47
3.1.2	Altering the state of a DCAM application	48
3.1.3	Querying the status of a DCAM application	49
3.1.4	Closing a DCAM application	49
3.2	Connection function	50
3.2.1	Connection setup: YOPNCON	50
3.2.1.1	Definition of the connection to be established	57
3.2.1.2	Connection request	63
3.2.1.3	Acceptance of a request	66
3.2.1.4	Connection setup - use of distribution codes	68
3.2.1.5	Linking up to a predefined connection	71
3.2.2	Querying entries on partners and connections	73
3.2.3	Rejecting a connection request	75
3.2.4	Changing the characteristics of a connection	75
3.2.5	Deleting a connection request	75
3.2.6	Closing a connection	76
3.3	Data communication function	77
3.3.1	Sending a message/data unit	78
3.3.2	Receiving a message/data unit	83
3.3.3	Combined sending and receiving	88
3.3.4	Canceling receive macros and changing the connection state CS/CA	88
3.3.5	Control distribution code assignment	88
3.4	Name assignment function	92
4	Support for virtual terminals	95
5	DCAM programs	97
5.1	ASSEMBLER programs	97
5.1.1	Macro calls and control blocks	97
5.1.2	Synchronous execution of DCAM calls	103
5.1.3	Asynchronous execution of DCAM calls	105
5.1.4	Feedback information	109
5.1.5	Asynchronous DCAM messages	111
5.1.6	Reentrant ASSEMBLER programs	115

5.2	COBOL programs	118
5.2.1	COBOL calls and data structures	118
5.2.2	Execute CALLs	123
5.2.2.1	Synchronous execution	123
5.2.2.2	Asynchronous execution	123
5.2.3	Feedback information	124
5.2.4	Reentrant COBOL programs	125
5.2.5	Using other system interfaces	128
5.3	Execution of a DCAM program: DCAM task	129
5.3.1	Starting a DCAM task	129
5.3.2	Terminating a DCAM task	131
5.3.3	Notes on programming	132
6	Appendix	135
6.1	DCAM calls	135
6.2	Limit values	138
6.3	Setting up a connection from a terminal	141
	Glossary	145
	Related publications	149
	Index	153

DCAM (BS2000)

Program Interfaces

Valid for
DCAM V11.0A

Comments... Suggestions... Corrections...

The User Documentation Department would like to know your opinion on this manual. Your feedback helps us to optimize our documentation to suit your individual needs.

Feel free to send us your comments by e-mail to:
manuals@ts.fujitsu.com

Certified documentation according to DIN EN ISO 9001:2000

To ensure a consistently high quality standard and user-friendliness, this documentation was created to meet the regulations of a quality management system which complies with the requirements of the standard DIN EN ISO 9001:2000.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright and Trademarks

Copyright © Fujitsu Technology Solutions GmbH 2010.

All rights reserved.

Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.



On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions. This document is a new edition of an earlier manual for a product version which was released a considerable time ago in which no changes have been made to the subject matter. Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions. Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format ...@ts.fujitsu.com. The Internet pages of Fujitsu Technology Solutions are available at [http://ts.fujitsu.com/...](http://ts.fujitsu.com/)