CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Transaction Capabilities (TC) allows Application Processes to exchange Components via Transaction Capabilities Application Part (TCAP) messages. Procedures described in this section specify the rules governing the information content and the exchange of TCAP messages between Application Processes.

1.1 Basic Guideline. To maximize flexibility in service architecture and implementation style, TCAP procedures restrict themselves to supporting the exchange of Components between Application Processes. Operation (Application) procedures are not part of TCAP.

1.2 Overview. Section 2 discusses addressing rules for TCAP messages. Section 3 describes TC procedure under normal (error-free, smooth running) conditions. Section 4 describes the methodology to cater for special message handling (handover etc.) situations. Section 5 details TC procedures under abnormal conditions.

2. ADDRESSING

TCAP messages will use any of the addressing options afforded by the Signalling Connection Control Part (SCCP). Assignment and use of Global Titles will be network specific. For internetwork transport, the Global Title used in one network may translate into the address of the access point to the next network for further translation. Alternatively, networks can exchange point codes and subsystem numbers for a particular service, thereby eliminating the need for intermediate translations.

Addressing within the Application Process may be accomplished via the use of Transaction IDs.

Addressing options available from the Application Service Part (ASP) are for further study.

3. NORMAL PROCEDURES

The interface between an Application Process and an Application Part is entirely implementation specific, i.e., it is not like a standardized primitive interface between two adjacent layers of the OSI Reference Model with specified services assigned to each layer. This section includes some discussions of the role of an Application Process as a user of the TCAP that can be viewed as outside the scope of the specifications of TCAP procedures and properly belonging to Q.771 and/or Q.772. Nevertheless, the discussions are included in this section to aid in understanding these specifications.

Also, when the selection of a parameter value required by a lower layer is not discussed in the TCAP procedures (e.g., Quality of Service), it is assumed that the value is specified by the Application Process, and TCAP simply passes the value down through its primitive interface. The same assumption applies to the parameters received from a lower layer through the primitive interface (e.g., calling party address) which are not required for TCAP functions.

3.1 Functional Grouping. The functions of the TCAP are grouped into two portions: Transaction and Component.

The Transaction Portion of the TCAP procedures provides an application level association over which Components are exchanged, i.e., it provides the means to identify a group of Components as

An asterisk '*' indicates a change from the CCITT Red Book, Vol. VI, that is specific to U.S. Networks.

A bar '|' indicates a change from Issue 1 of Bell Communications Research Specification of Signalling System Number Vol. 1 and 2.

belonging to a particular transaction. The Component portion of the TCAP procedure provides an Application Process with the capability to invoke an operation at a remote Application Process and receive the responses.

The fields of a TCAP message correspond to this functional grouping of the features of the TCAP procedure.

3.2 Transaction Portion.

3.2.1 Connectionless Network Services (No Application Services Part Functions Required). The transaction portion of TCAP procedures identifies each TCAP message and, therefore, all the contained Components as belonging to a particular Application Process transaction. In addition, the procedures allow association of separate TCAP messages in any direction to an Application Process transaction.

Transaction association within an Application Process serves two functions: It allows linking of a query with its response. In that function, transaction association serves a role like that of Component correlation. In its other function, transaction association identifies the context to help interpret a broader group of Components contained in one or more TCAP messages.

Transaction IDs identify an Application Process transaction, while the Package Types indicate the sending end's view on establishing and maintaining a TCAP transaction.

3.2.1.1 Actions at the Initiating End. When an Application Process has to send one or more Components in a TCAP message to another Application Process but does not need to enter into a TCAP transaction, the Unidirectional Package type is specified to TCAP. Use of Unidirectional Package Type implies that Component correlation is not applicable. No Transaction ID is included in a TCAP message of the Unidirectional Package Type. In all other cases, a TCAP transaction is established.

To help clarify the discussion of TCAP transaction message exchange, the sending node of the first TCAP message is labelled node 'A', and the receiving node is labelled node 'B'.

When an Application Process at node 'A' wishes to initiate a TCAP transaction with an Application Process at node 'B', the first message is one of the two Query Package Types (With or Without Permission to release). The Application Process at node 'A' selects and assigns to the TCAP message of the Query Package Type a Transaction ID value for the originating Transaction ID field. This Transaction ID value, when included in any future message from node 'A' as the originating or in a message to node 'A' as the Responding Transaction ID, identifies the Application Process transaction, i.e., that the messages and, therefore, the contained Components belong to the same Application Process transaction from the perspective of node 'A'.

When the Application Process at node 'A' initiating a TCAP transaction anticipates sending more Components which it would like the Application Process at node 'B' to treat as part of the same transaction, the Application Process at node 'A' specifies a Query Without Permission (to release) Package Type.

When the TCAP transaction initiating Application Process at node 'A' anticipates the converse of that described above, it sends a TCAP message of the Query With Permission (to release) Package Type.

3.2.1.2 Actions at the Receiving End. No response is required on the receipt of a message of the Unidirectional Package Type.

In response to a TCAP message of the Query Without Permission (to release) Package Type, the Application Process at node 'B' should establish an Application Process transaction from its perspective by responding with a message of one of the two Conversation Package Types (With or Without Permission to Release).

A message of the Conversation Package Type from node 'B' includes a responding Transaction ID value which is identical to, i.e., a reflection of, the originating Transaction ID value of the TCAP transaction initiating message from node 'A'. In addition, a message of the Conversation Package Type is also assigned an originating Transaction ID by node 'B'. This Transaction ID value, as in the corresponding situation in node 'A', when included in any future message, identifies the Application Process transaction from the perspective of node 'B'.

In responding to a message of the Query With Permission (to release) Package Type, the Application Process at node 'B' decides whether or not to establish an Application Process transaction from its perspective. If the Application Process at node 'B' does want to establish an Application Process transaction from its perspective, it responds with a message of one of the two Conversation Package Types, and the procedure described in the last paragraph applies. Otherwise, it responds with a message of the Response Package Type. A message of the Response Package Type includes only one Transaction ID - a responding Transaction ID assigned in the same manner as described in the last paragraph.

3.2.1.3 Conversation Mode. Once the Application Process at node 'B' has sent a message of one of the two Conversation Package Types, all future messages in either direction remain one of the two Conversation Package Types until the TCAP transaction is to be terminated.

3.2.1.4 Permission or not to Release in the Conversation Mode. An Application Process sends a message of the Conversation With Permission Package Type, when, for the present TCAP transaction, the Application Process 1) has completed responding to all received Components, and 2) does not anticipate sending any new Components without further interactions with the peer Application Process. Otherwise, a message of the Conversation Without Permission Package Type is sent.

The permission given by one Application Process to another Application Process to release a TCAP transaction at the conversation mode is not a request to release, it is only a permission to release. It is the Application Process receiving the permission which has to decide whether or not to release the TCAP transaction. It may send several more messages to either complete responding to previously received Components or send new Components over the TCAP transaction before releasing. Also, the permission to release given by an Application Process can be revoked. Based on future events, the Application Process which gave the permission is allowed to revoke it by sending a message of the Conversation Without Permission (to release) Package Type.

3.2.1.5 Termination of TCAP Transaction. The usual way to terminate a TCAP transaction is for the Application Process which has received the permission to release from the remote Application Process, to send, when it chooses, a message of the Response Package Type.

In addition, by prearranged agreements, a TCAP transaction may be terminated at the discretion of the Application Process at both ends without sending or receiving an explicit Response message. Application Processes will inform the respective TCAPs that the transaction has been terminated.

In special situations, independent of whether an Application Process has received the permission to release from the other end or not, the Application Process can terminate the TCAP transaction by sending a message of the Response Package Type.

Figures 1/Q.774 and 2/Q.774 depict examples of exchanges of TCAP messages between two Application Processes. Figure 3/Q.774 depicts another example of message exchange to illustrate how a TCAP transaction can be terminated by one end, while the Application Process transaction is continued by the other end.

3.2.1.6 Application Process Transaction vs Logical Connection. For clarification of Application Process transaction and its local significance, a comparison with a logical connection (SCCP type) is useful. A group of Components transferred in one or more TCAP messages with the same Transaction ID assigned by say an Application Process at a node 'A' are viewed as belonging to a single Application Process transaction by the Application Process at node 'A'. At the other end, from the perspective of an Application Process at a node 'B', the Components may be viewed as forming multiple groups belonging to separate Application Process transactions, each identified by a Transaction ID assigned by 'B'. In the context of SCCP, this would be like associating multiple destination reference numbers to a single source reference number. While the TCAP procedures does allow emulating a logical connection by viewing the exchanged components as belonging to a single Application Process transaction from the point of view of either of the ends by assigning only one Transaction ID at each end, the TCAP procedure does not insist on it.

Another example of the flexibility of the TCAP procedure for Application Process transaction is illustrated by point to multipoint communications. An Application Process can send 'n' components, one each to 'n' other Application Processes. While each remote Application Process views the Component exchange as an independent Application Process transaction, TCAP procedure allows the sending Application Process the flexibility to view the exchanges as 'n' different Application Process transactions by assigning 'n' different originating Transaction IDs, or as one Application Process transaction by assigning a single originating Transaction ID to all the messages.

3.2.2 Connection oriented Network Services. This area is for further study.

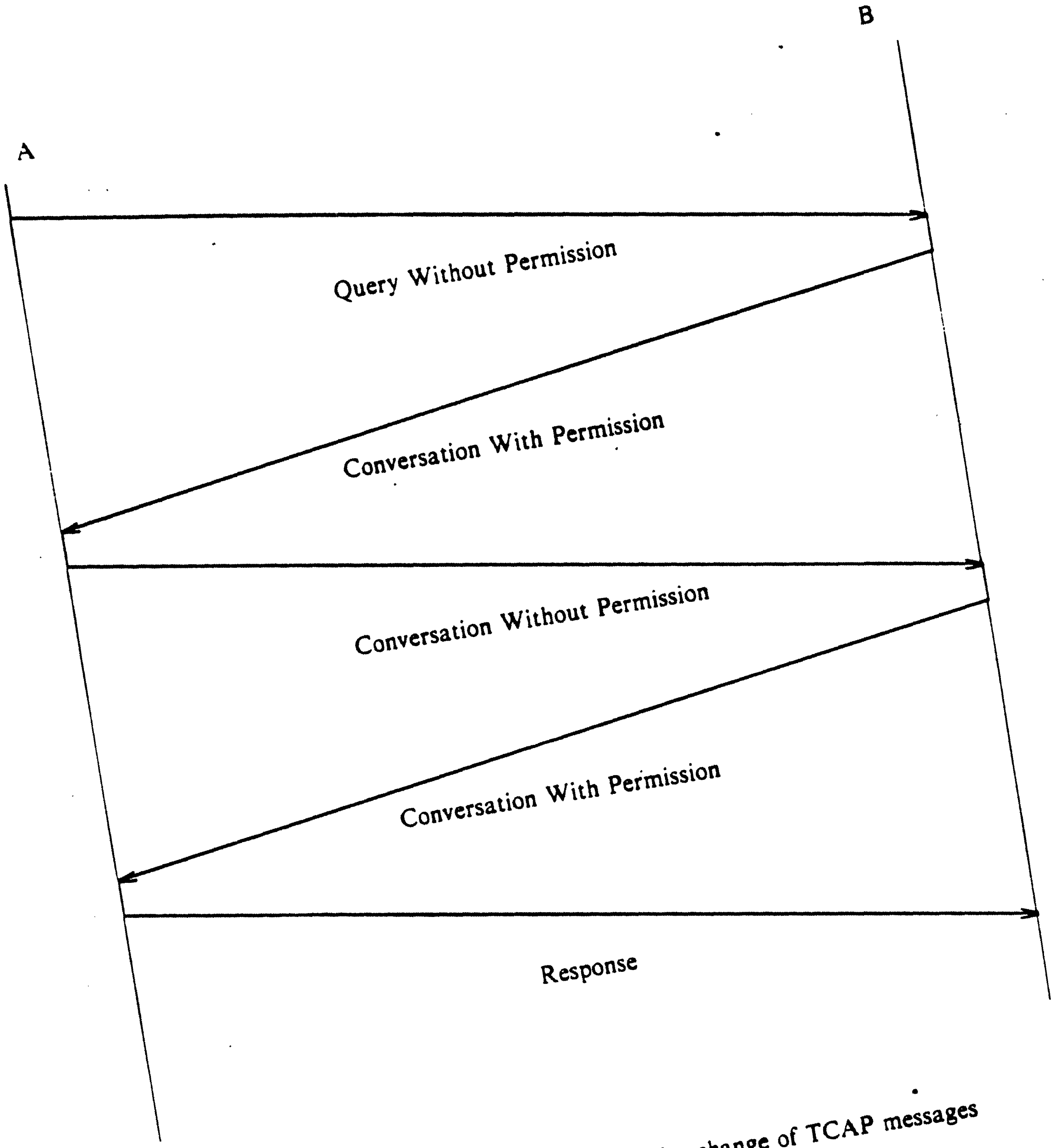

A
B
Query Without Permission
Conversation With Permission
Conversation Without Permission
Conversation With Permission
Response

Figure 1/Q.774. A simple example of exchange of TCAP messages

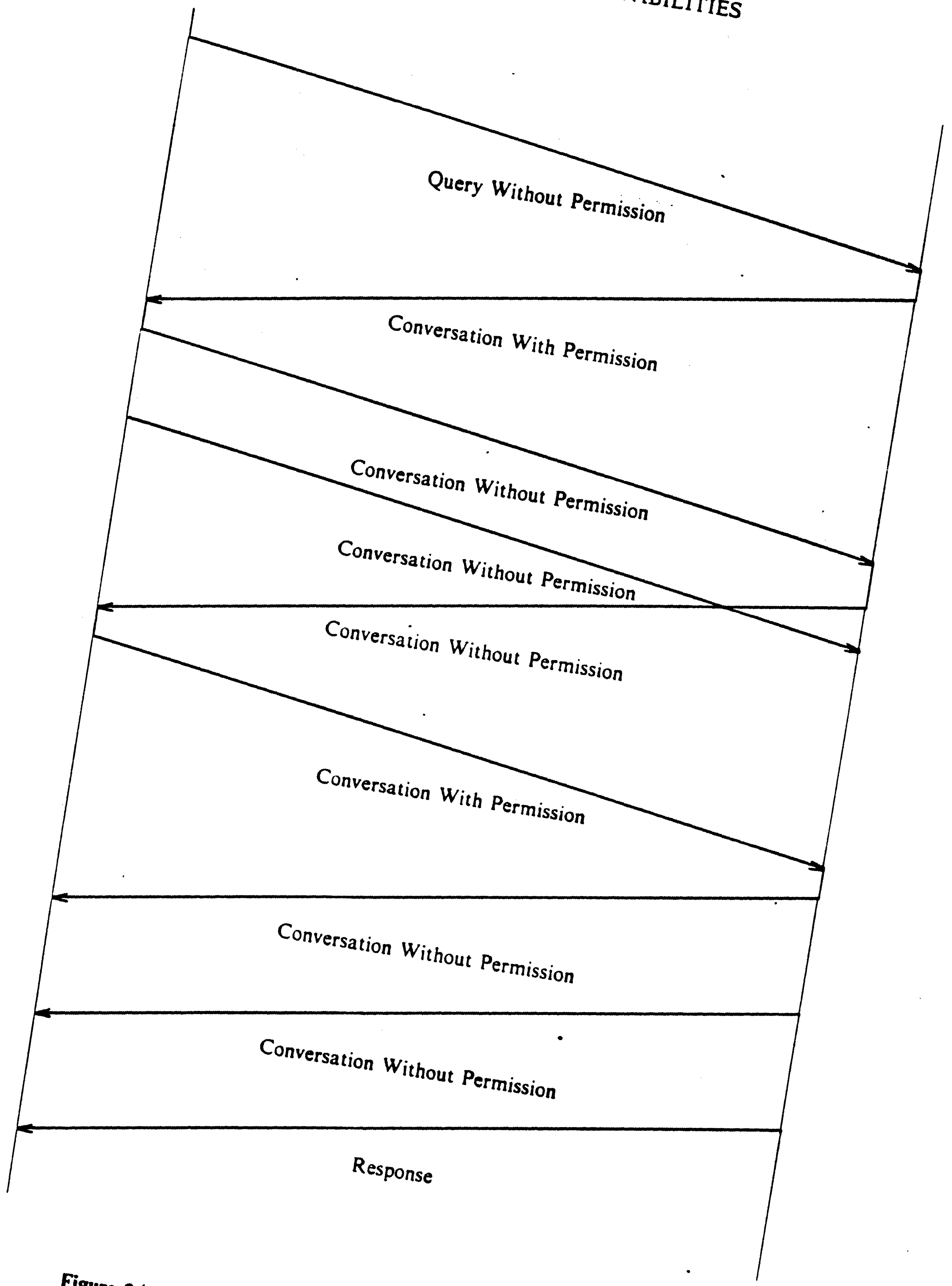
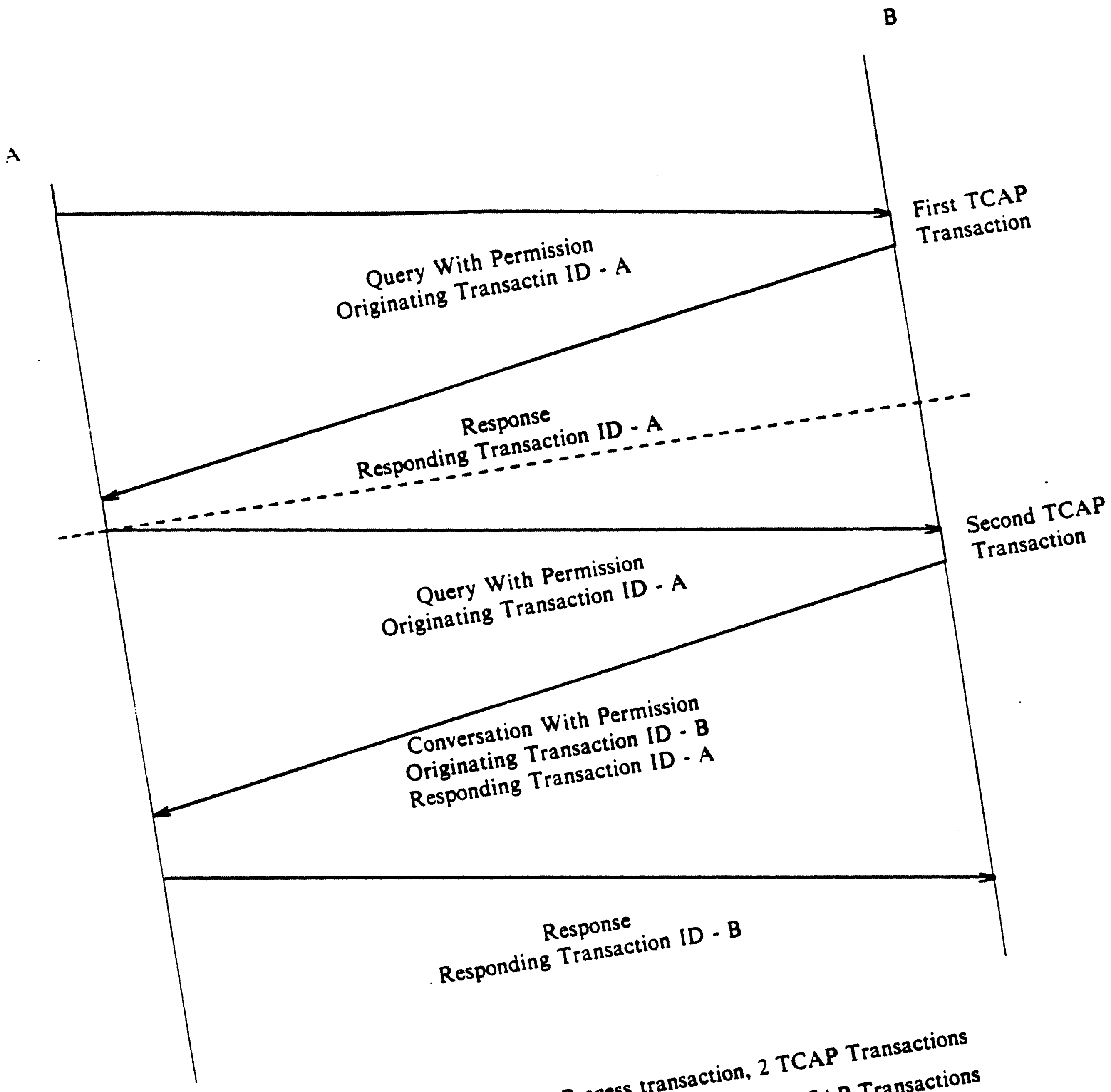

Query Without Permission
Conversation With Permission
Conversation Without Permission
Conversation Without Permission
Conversation Without Permission
Conversation With Permission
Conversation Without Permission
Conversation Without Permission
Response

Figure 2/Q.774. A complex example of exchange of TCAP messages


B
A
First TCAP Transaction
Query With Permission
Originating Transactin ID - A
Response
Responding Transaction ID - A
Second TCAP Transaction
Query With Permission
Originating Transaction ID - A
Conversation With Permission
Originating Transaction ID - B
Responding Transaction ID - A
Response
Responding Transaction ID - B

From A's perspective, this is 1 Application Process transaction, 2 TCAP Transactions
From B's perspective, this is 2 Application Process transactions, 2 TCAP Transactions

Figure 3/Q.774. An Example of Multiple TCAP Transactions

3.3 Component Portion. A TCAP message can carry multiple Components: each Component corresponds to a single OPDU of X.410 as extended for TCAP. Under normal conditions, Application Processes send and/or receive Invoke and Return Result Components only.

An Application Process provides to TCAP all the elements needed to construct a Component. An Invoke Component includes a single operation and the parameters (argument) necessary to perform the operation. Therefore, for an Invoke Component, the Application Process provides to TCAP the name of the Operation, the name of the parameters and the parameter values. Similarly, for a Return Result Component, the Application Process provides the parameters (results) to TCAP to be contained in the Component.

An Application Process need not wait for one operation to complete before invoking another. At any instant in time, an Application Process may have any number of operations in progress at another Application Process.

The other information provided by an Application Process to TCAP to formulate a Component relates to Component correlation and Component states.

In TCAP, it is permissible to respond to an Invoke with another Invoke, as well as a Return Result. It is also permissible in TCAP to respond to an Invoke by any number of Invokes and Return Results in one or more TCAP messages. In addition a Return Result Component may be used when no explicit Invoke Component has been sent (e.g., to periodically report status information).[1]

A TCAP message can contain any mix of Components: initial Invoke Component, Invoke Component which is responding to a previous Invoke Component, and Return Result Component (Return Error and Reject Components can also be included in TCAP messages and will be discussed under abnormal procedures in Section 5, Q.774).

Figure 4/Q.774 depicts an example of an exchange of Components in a single TCAP transaction. The Component identifiers are shown within parentheses; the Invoke ID is followed by the Correlation ID, if used. Also shown explicitly is whether or not a responding Component is the last of the responses.

Assignment of Component Identifiers by an Application Process

Invoke ID: When the sending Application Process needs to do so, it selects and assigns an Invoke ID to the Invoke Component. When an Application Process sends an Invoke with a Correlation ID, the Invoke must carry an Invoke ID.

Correlation ID: A Correlation ID needs to be included in a Return Result Component, as well as in an Invoke Component if it is responding to a previous Invoke which included an Invoke ID. In either case, the Correlation ID is identical to, i.e., a reflection of, the Invoke ID of the Component being responded to.

To further clarify, an Application Process can specify for TCAP:

a) Invoke ID for an Invoke Component (non responding),
b) Invoke ID and Correlation ID for an Invoke Component (responding),
c) Correlation ID for a Return Result Component.

1. These are extensions to X.410 which implies that a single Return Result may respond to an Invoke.

Assignment of Component States by an Application Process

For an Invoke Component, the Application Process specifies to TCAP whether or not a reply is required to the Component.

When a Component (whether an Invoke or a Return Result) is responding to a previous Component, i.e., when a Component includes a Correlation ID, the Application Process specifies to TCAP whether or not the Component is the last response to the invoking Component.

Maintenance of Invoke IDs

The Invoke ID distinguishes the associated operation from any number of other operations the invoking Application Process may have in progress at the invoked Application Process. The invoking Application Process may not reuse an Invoke ID that was previously assigned to an operation for which it expects but has not yet received a complete response. Similarly, the invoked end maintains the received Invoke ID (to be reflected as the Correlation ID) until it has completed responding. Figure 4/Q.774 depicts the above concepts.

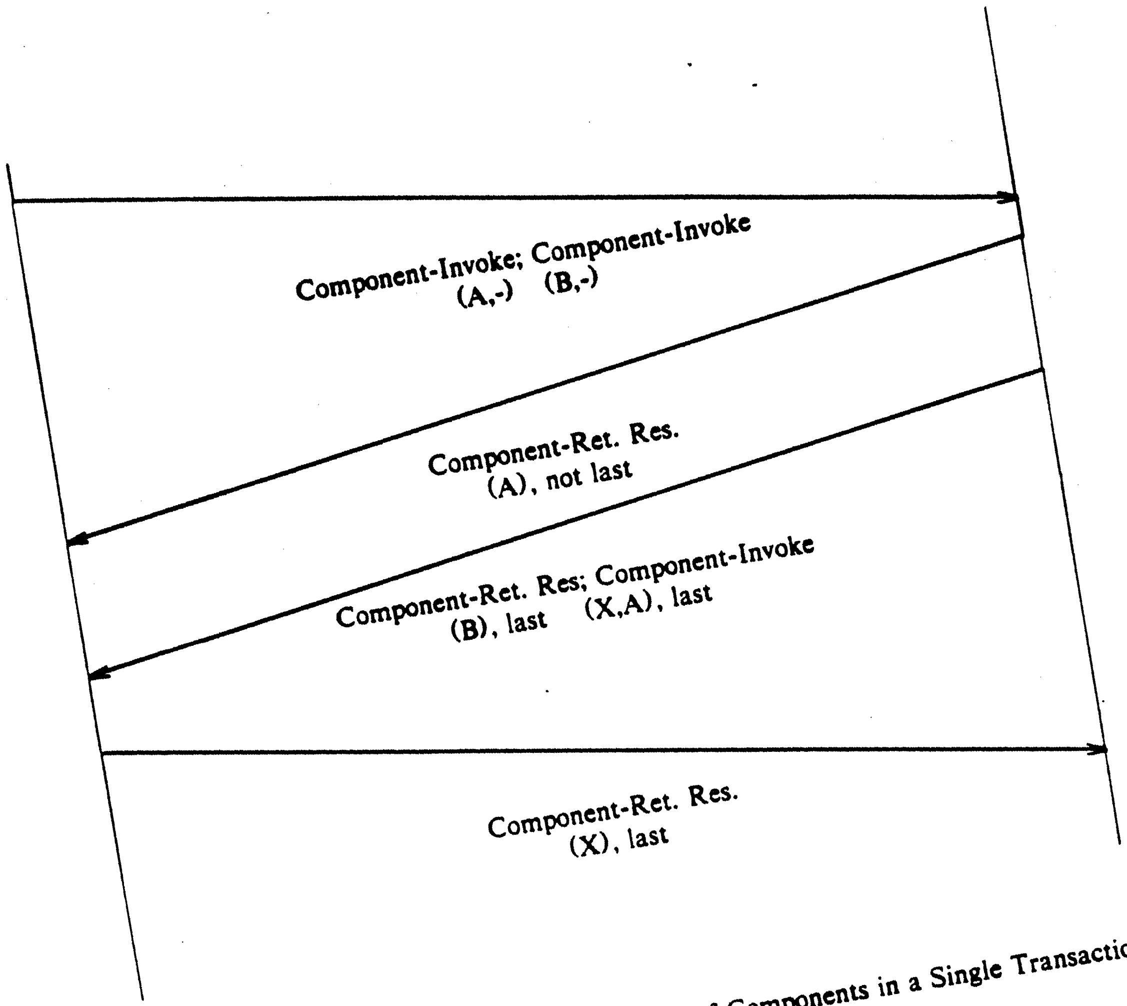


Figure 4/Q.774. An Example of an Exchange of Components in a Single Transaction

4. SPECIAL PROCEDURES

4.1 Handover. At some time during the exchange of TCAP Messages between Application Processes at two nodes (A and B), the Application Process at one node (B) may decide that it wishes to transfer a Component or series of Components to an Application Process at node C (See Figure 5/Q.774). A handover is defined to be such a transfer of responsibility to process the Component. This handover may be permanent (i.e., for the rest of the transaction) or temporary. The handover can occur before or after any responses from B to A.

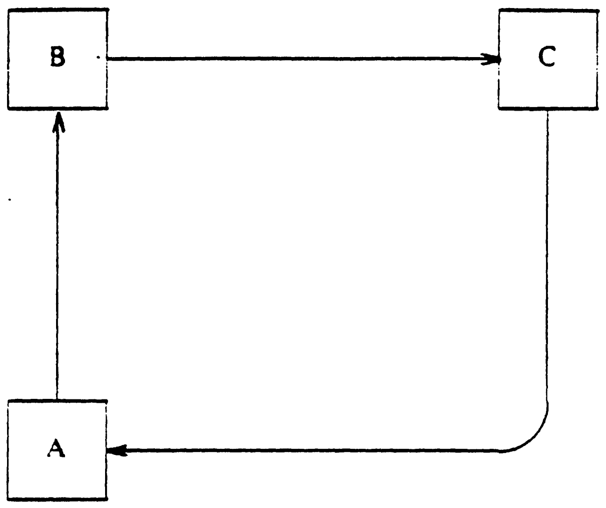


Figure 5/Q.774. Handover

4.1.1 Handover Origination.

4.1.1.1 Temporary Handover. When the Application Process at B determines that a temporary handover is necessary, it sends a TCAP Message to C that contains an Invoke Component specifying the Temporary Handover operation followed by the other Components that B wishes C to process. The Invoke Component (with the Temporary Handover operation) specifies, as parameters, the Transaction ID (of B), SCCP Calling Party Address (of B), and Package Type that C should use in its message to A.

4.1.1.2 Permanent Handover. When the Application Process at B determines that a permanent handover is necessary, it sends a TCAP Message to C that contains the Components that B wishes C to process. The SCCP Calling Party Address and the Transaction ID are set to that of A, so that C can respond directly to A. In effect, the application at B is performing relaying at the application level.

4.1.2 Handover Receipt.

4.1.2.1 Temporary Handover. The Application Process at C recognizes the receipt of a temporary handover by the presence of the Invoke Component specifying Temporary Handover operation. After processing the Components received from B, the Application Process at C responds to A using the Package Type and Responding Transaction ID as specified in the Temporary Handover Invoke Component . The SCCP Calling Party Address is set to that of B. The Application Process at C can then remove all references to the Components and the handover.

4.1.2.2 Permanent Handover. The Application Process at C will not be able to differentiate a TCAP Message that is a result of a permanent handover from one that was directly routed to it. Normal procedures apply to this message.

4.1.3 Abnormal Conditions in Handover. The handling of abnormal conditions with regard to handover procedures (e.g., permanent handover back to the original node) is for further study.

4.2 Acknowledgement of Receipt of Components. (for further study)

5. ABNORMAL PROCEDURES

5.1 Connectionless Network Service.

5.1.1 General. This section describes the procedures and messages needed to detect, report, and recover from abnormal conditions associated with TCAP or the supported Application Process. Application Process dependent detection and recovery procedures are not considered part of TCAP.

5.1.2 Introduction. The Application Process sending a TCAP message is responsible for insuring that the message is precise and correct. The Application Process which should receive the message is responsible for detecting abnormal conditions. It is also responsible for reporting abnormal conditions to the Application Process causing the abnormal condition. Abnormal conditions may also be reported locally to maintenance. In addition to triggering recovery procedures, reporting abnormal conditions allows program and data errors to be more easily identified. The Application Process is responsible for any reattempt recovery procedure.

5.1.3 Abnormal Conditions. Abnormal conditions can be divided into three categories based on where the error occurred - protocol, application, and end user.

5.1.3.1 Protocol Errors. Protocol errors are caused by incorrect TCAP messages. These errors are detected by either TCAP or the Application Process. For example, this category includes but is not limited to:

- Unrecognized Package Type or Component Type
  Package Type or Component Type not defined in Q.773
- Unrecognized operation
  Operation not defined for this Application Process
- Unrecognized responding Transaction ID or Correlation ID
  No such transaction or operation in progress

5.1.3.2 Application Errors. Application errors are caused by violations of Application Process procedures or unavailability of network resources. For example, this category includes but is not limited to:

- Unexpected sequence of Components
  Components do not follow application script
- Unexpected data value
  Data value not defined for this operation and application
- Unavailable network resource (e.g., announcement, call register)
  Shared network resource temporarily not available
- Missing customer record
  Cannot find customer record for this application
- Reply overdue
  Reply to invoke exceeds application performance requirements (implicitly agreed to or explicitly exchanged via an invoke)

5.1.3.3 End User Abnormalities. End user abnormalities are caused by the end user violating the correct Application Process procedure, even though the error is within the definition of the Application Process. For example, this category includes but is not limited to:

- Caller abandonment
  Caller hangs up prematurely
- Improper caller response
  Improper information input during caller participation phase

5.1.4 Detection. Detection of errors is performed in the following sequence:

1. Protocol Errors
2. Application Errors
3. End User Abnormalities

Therefore, detecting an application error means that there are no protocol errors, and detecting an end user abnormality implies that there are no protocol or application errors.

Detection of protocol errors is governed by TCAP abnormal procedures. However, some protocol errors may actually be detected by the Application Process. Detection of application and end user errors are entirely Application Process dependent. Each Application Process defines those application and end user abnormal conditions it will detect and those it will report.

5.1.5 Reporting. Abnormal conditions are reported to the Application Process causing the error using the Reject, Return Error, and Return Result Components defined in Q.772.

5.1.5.1 Reject. Protocol errors in both the Transaction portion and Component portion of a TCAP message are reported using the Reject Component. The Reject Component reports the receipt and rejection of an incorrect Package or a Component. It is sent in eventual response to an incorrect Package, or a Component whose type is other than Reject.

Reject: [ID, Problem]

When a rejected Invoke includes an Invoke ID or a rejected Return Result (or Return Error) includes a Correlation ID, this ID is reflected in the Reject Component.

Problems are divided into five categories - Transaction Portion, General, Invoke Component, Return Result Component, Return Error Component.

The following Transaction Portion problems are reported:

1. Unrecognized Package Type
   The Package Type has not been defined
2. Incorrect (Mistyped) Transaction Portion
   An unexpected or undefined identifier was received within the Transaction Portion
3. Badly structured Transaction Portion
   A fundamental encoding problem (e.g., bad length)
4. Unrecognized Transaction ID
   The received Transaction ID does not reflect a transaction currently in progress
5. Permission to release problem (for further study)

The following General problems are reported:

1. Unrecognized Component Type
   The component type has not been defined
2. Incorrect (Mistyped) Component Portion
   An unexpected or undefined indicator was received within the Component Portion
3. Badly structured Component Portion
   A fundamental encoding problem (e.g., bad length)

The following Invoke Component-specific problems are reported:

1. Duplicate Invoke ID (when requested by the Application Process)
   An Invoke ID is received which has already been assigned to another Operation in progress
2. Unrecognized Operation Code
   The Operation Code has not been defined by the Application Process
3. Incorrect (Mistyped) Parameter
   An unexpected or undefined Parameter was received
4. Unrecognized Correlation ID
   The received Correlation ID does not reflect an Operation currently in progress

The following Return Result Component-specific problems are reported:

1. Unrecognized Correlation ID
   The received Correlation ID does not identify an Operation currently in progress
2. Unexpected Return Result
   The invoked operation does not report success
3. Incorrect (Mistyped) Parameter
   An unexpected or undefined parameter was received

The following Return Error Component-specific problems are reported:

1. Unrecognized Correlation ID
   The received Correlation ID does not reflect an operation currently in progress
2. Unexpected Return Error
   The Return Error Component does not report failure of the invoked operation
3. Unrecognized error
   The reported error has not been defined by the Application Process
4. Unexpected error
   The reported error is not applicable to the invoked operation
5. Incorrect (Mistyped) parameter
   An unexpected or undefined parameter was received

Return Result and Return Error - specific problems cannot always be correlated in the Component portion of TCAP because Return Result Component and Return Error Component Correlation IDs are not normally retained, only reflected. Invokes which do not carry an Invoke ID have a similar problem. Correlation in the Transaction portion of TCAP may or may not be possible depending on the number of Transaction IDs used. Even without correlation in the Transaction or Component portion of TCAP, tabulation of reported abnormal conditions can be useful for detecting program and data errors.

5.1.5.2 Return Error. The Return Error Component reports the unsuccessful completion of an operation. It is sent in eventual response to an Invoke Component if the latter is correct, the operation is one that reports failure only or both success and failure, and the operation fails. The Application Process defines which application errors and end user abnormalities are considered failures from the perspective of the operation.

Return Error: [ID, Error, Parameter]

If the Invoke Component includes an Invoke ID or the Return Result Component includes a Correlation ID, this ID is reflected in the Return Error Component. The error element specifies the application error being reported.

The parameter (variable data) specifies the invalid data value, if applicable.

5.1.5.3 Return Result. The Return Result Component reports the successful completion of an operation. The Application Process defines which application errors and end user abnormalities are considered abnormal conditions from an Application Process perspective, but successful completions from the perspective of the operation.

Return Result: [ID, Parameters]

The ID element (as in 5.1.5.2) identifies the operation whose success is being reported.

A parameter specifies the end user error type.

5.1.6 Recovery. Recovery from errors is Application Process dependent. In addition to improving the management of Transaction IDs and other resources associated with a particular transaction, abnormal condition reports can be used to identify errors in stored data.

5.2 Connection-oriented. This area is for further study.

6. STATE TRANSITION DIAGRAMS

6.1 Overview. Section 6 contains the specification of Transaction Capabilities Application Part described in this Recommendation, in the form of state transition diagrams according to the CCITT Specification and Description Language (SDL). The SDL is specified in CCITT Recommendations Z.100 to Z.104. The following list summarizes the TCAP state transition diagrams:

- TCAP Functional Block Diagram: Figure 6/Q.774.
- Transaction Portion: Figure 7/Q.774.
- Transaction Portion Component Portion Interface: Figure 8/Q.774.
- Component Portion: Figure 9/Q.774.

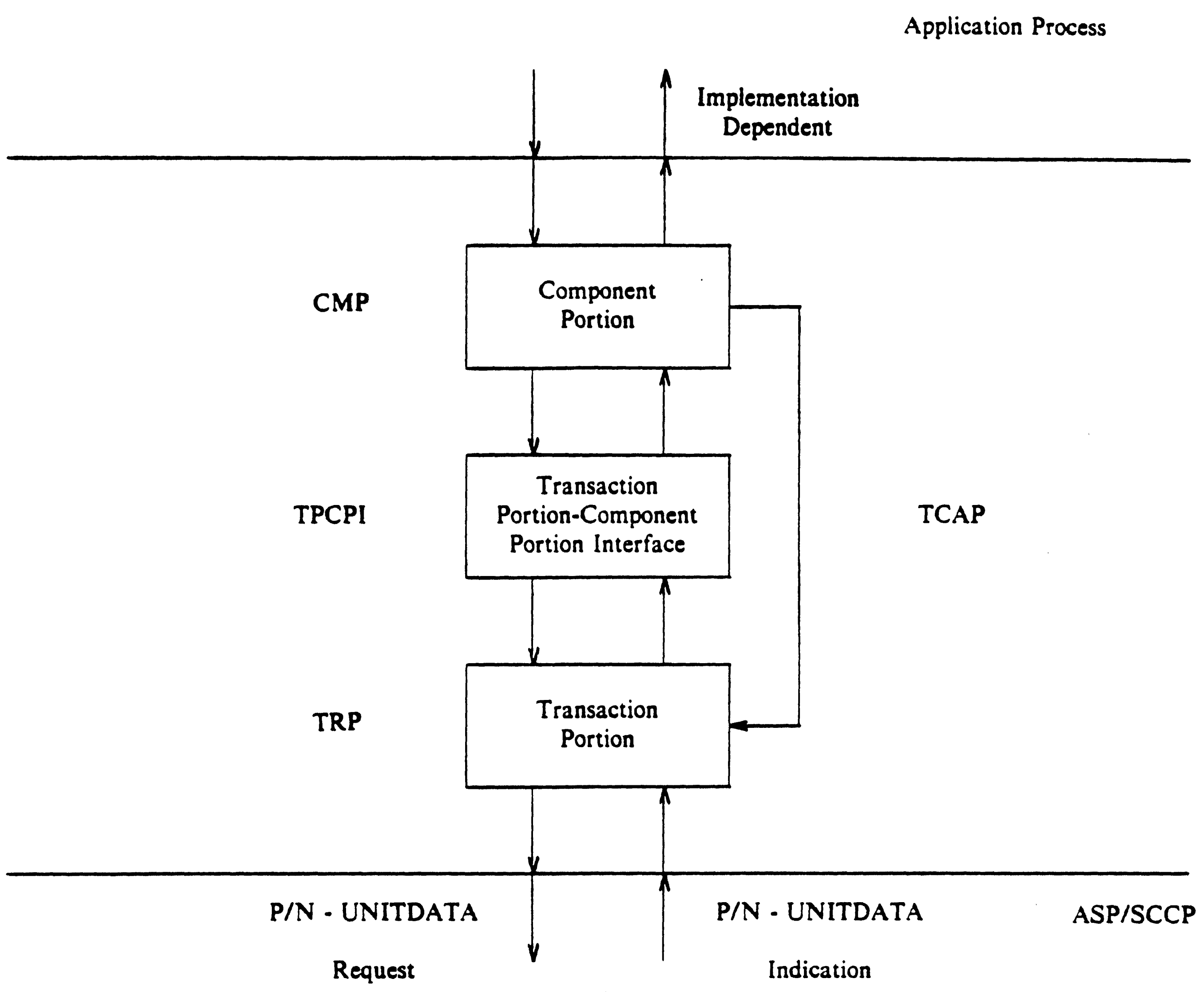
The detailed functional breakdown shown in the following diagrams is intended to illustrate a reference model and to assist interpretation of the text in earlier sections. The state transition diagrams are intended to show precisely the behavior of the signalling system under normal and abnormal conditions as viewed from a remote location. It must be emphasized that the functional partitioning shown in the following diagrams is used only to facilitate understanding of the system behavior and is not intended to specify the functional partitioning to be adopted in a practical implementation of the signaling system.

The State Transition Diagrams describe a half duplex interchange of TCAP messages and components. Extension of the State Transition Diagrams to cover other cases is for further study.

6.2 Abbreviations used in the state transition diagrams:

+	=	With
-	=	Without
ASP	=	Application Service Part
CMP	=	Component Portion
CON	=	Conversation
PER	=	Permission
QRY	=	Query
RID	=	Response ID
RR	=	Return Result
RE	=	Return Error
TID	=	Transaction ID
TRP	=	Transaction Portion
TPCPI	=	Transaction Portion - Component Portion Interface
UNI	=	Unidirectional

In the case of states labeled, for example "CON(+) Received", the abbreviation means that the protocol is in a Conversation state and a Package Type of Conversation with Permission to Release has been received: or "CON(-) Sent" means that a Package Type of Conversation Without Permission has been sent.



CMP = Component Portion

TRP = Transaction Portion

TPCPI = Transaction Portion-Component Portion Interface

Figure 6/Q.774. TCAP Functional Blocks

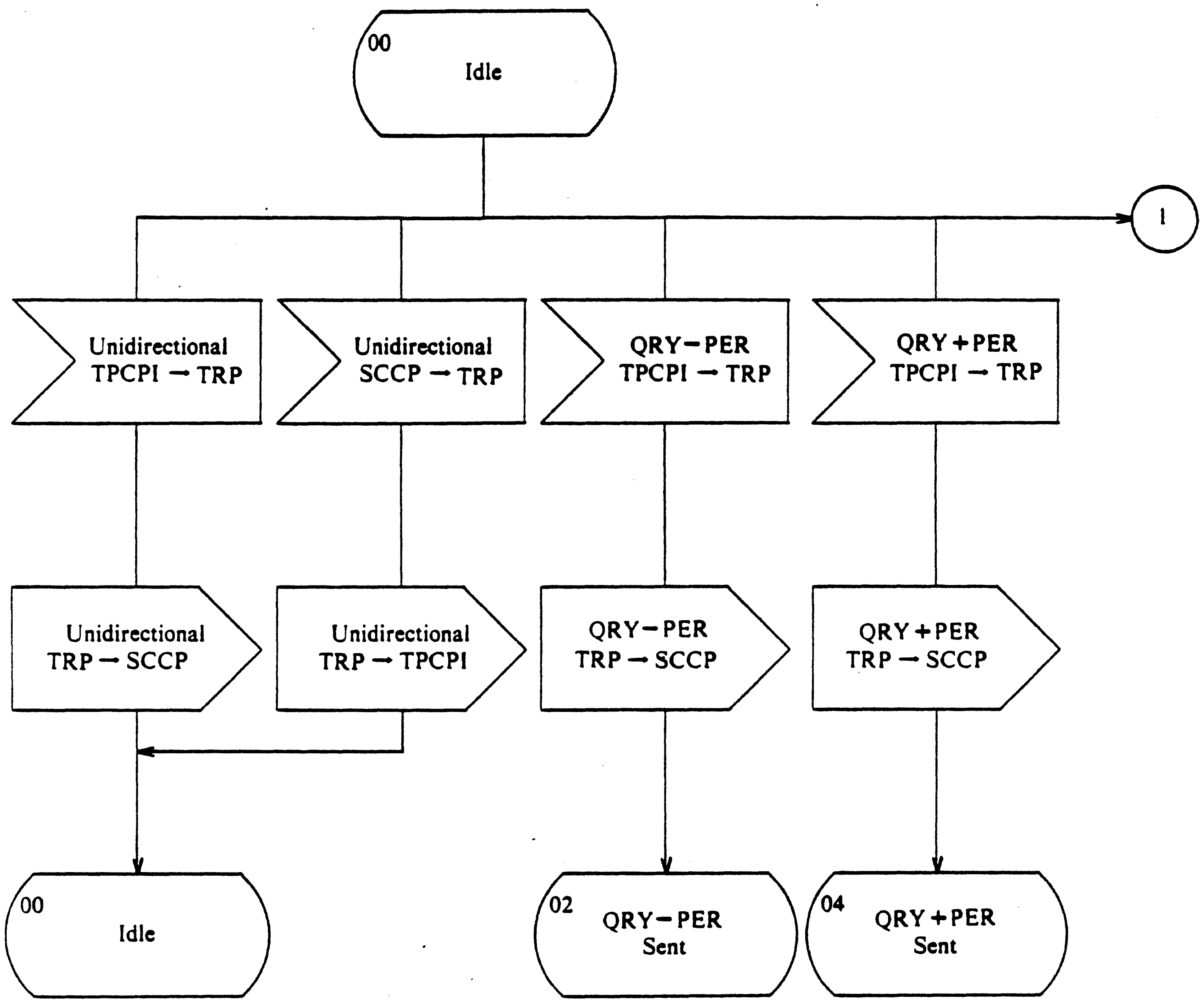


Figure 7/Q.774. Transaction Portion (Sheet 1 of 10)

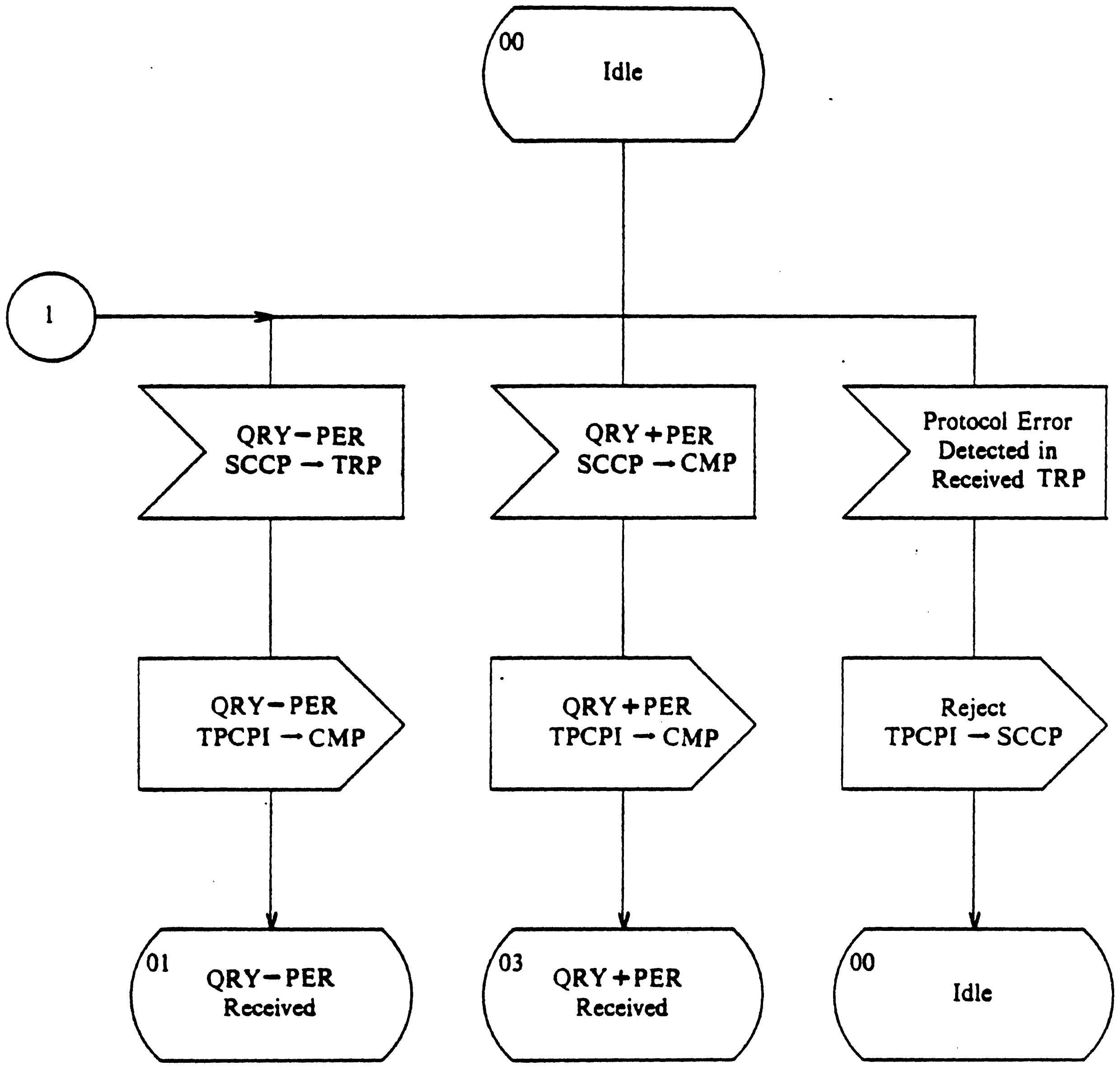


Figure 7/Q.774. Transaction Portion (Sheet 2 of 10) * |

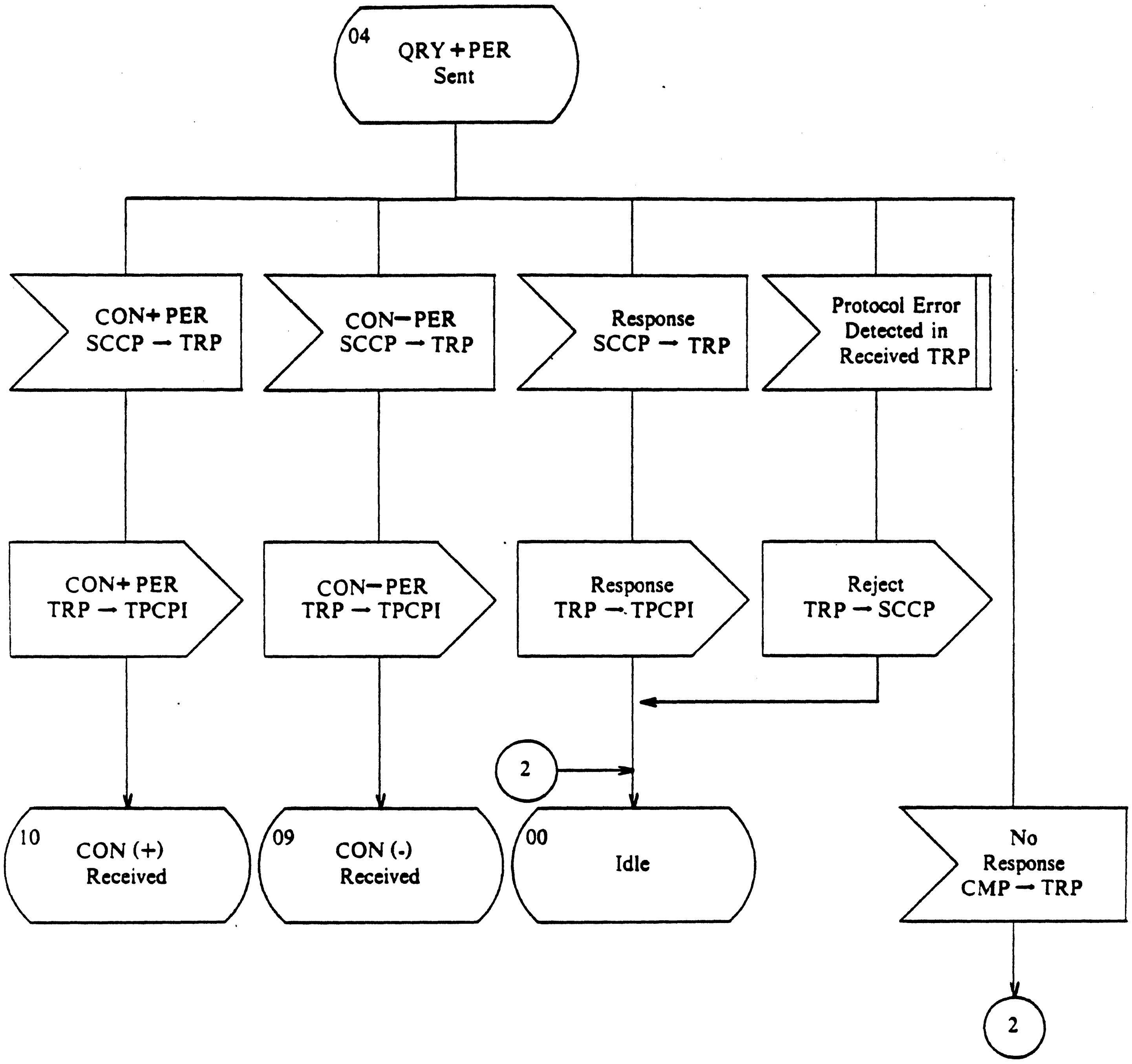


Figure 7/Q.774. Transaction Portion (Sheet 3 of 10)

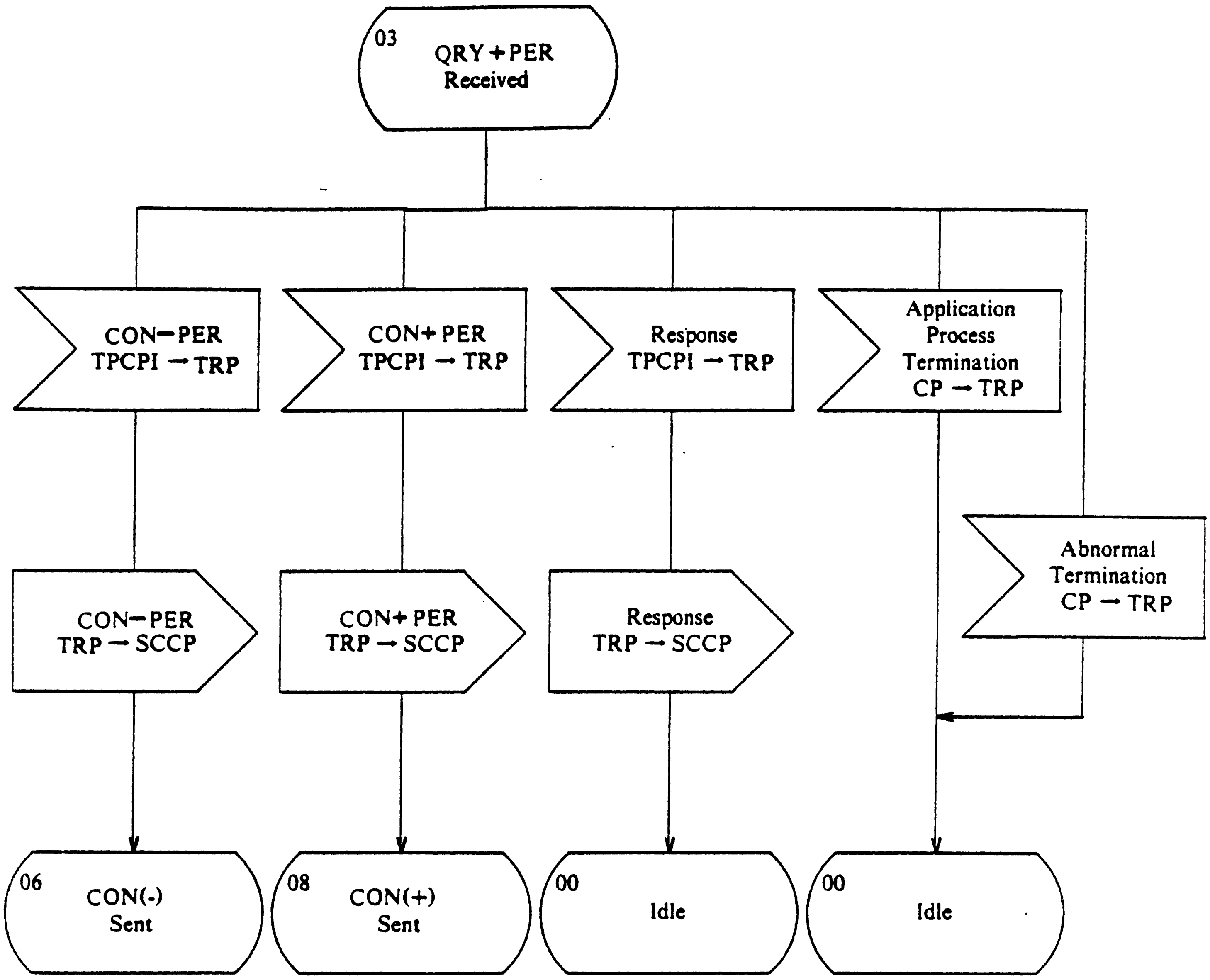


Figure 7/Q.774. Transaction Portion (Sheet 4 of 10)

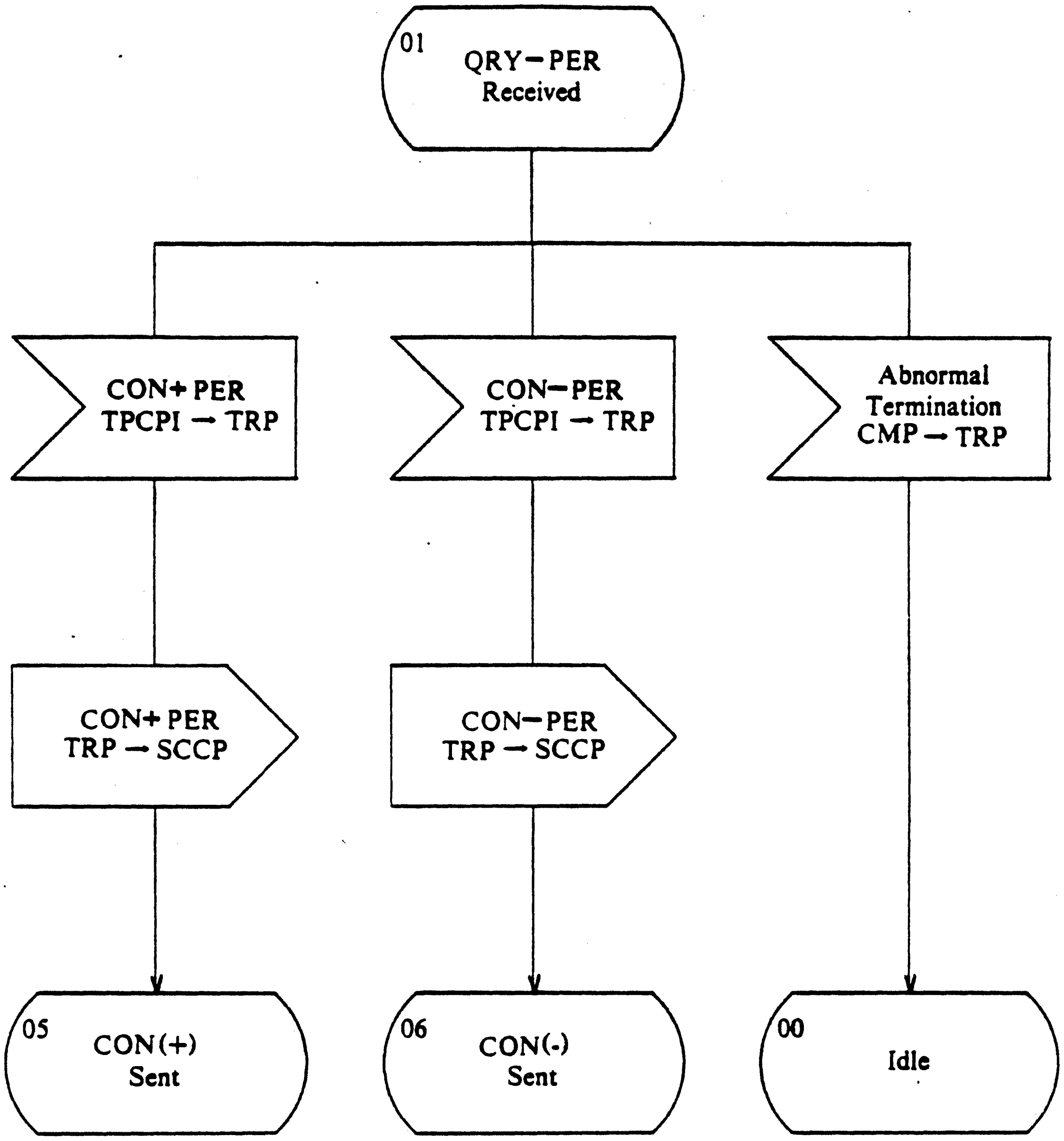


Figure 7/Q.774. Transaction Portion (Sheet 5 of 10) * |

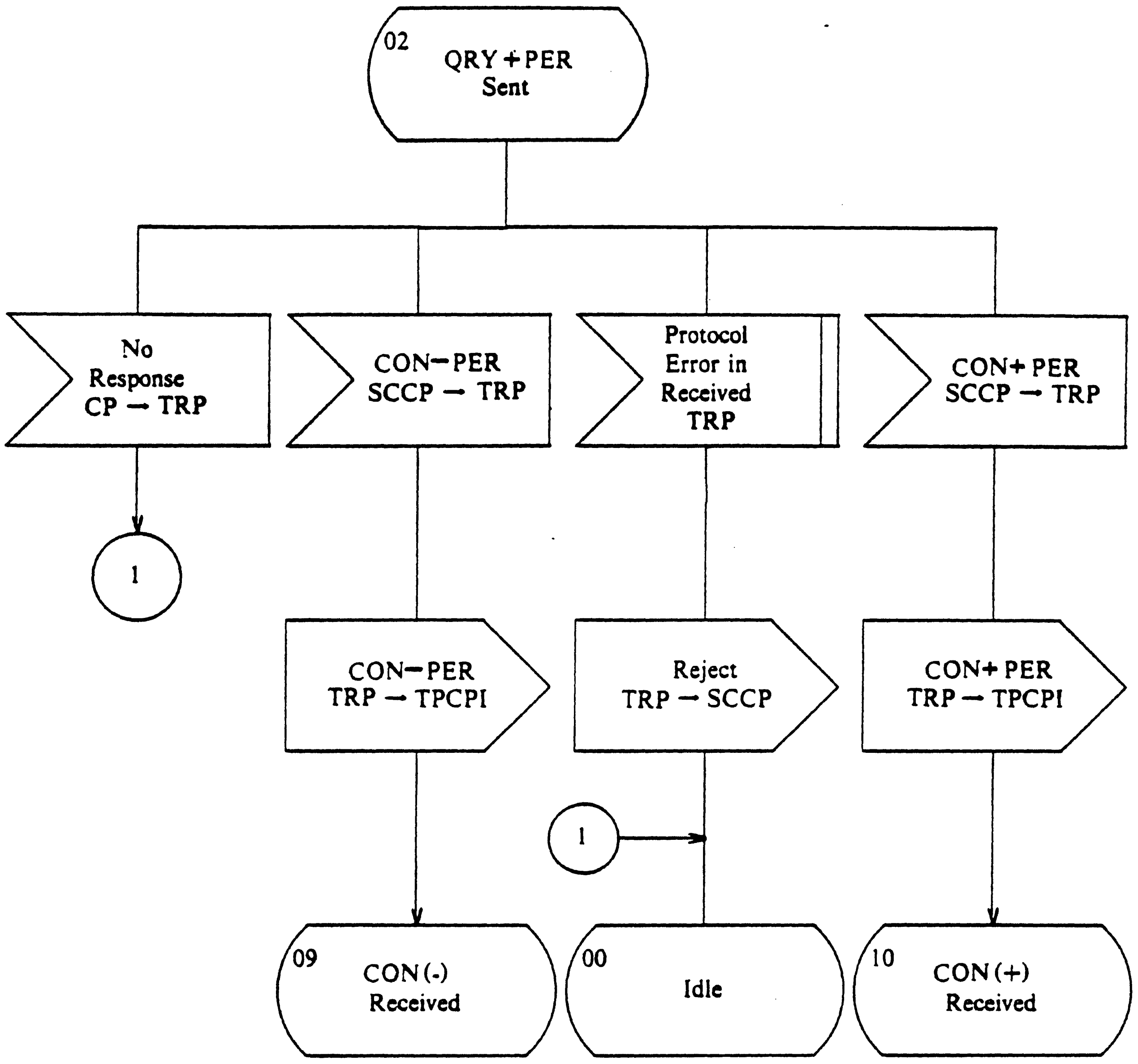


Figure 7/Q.774. Transaction Portion (Sheet 6 of 10)

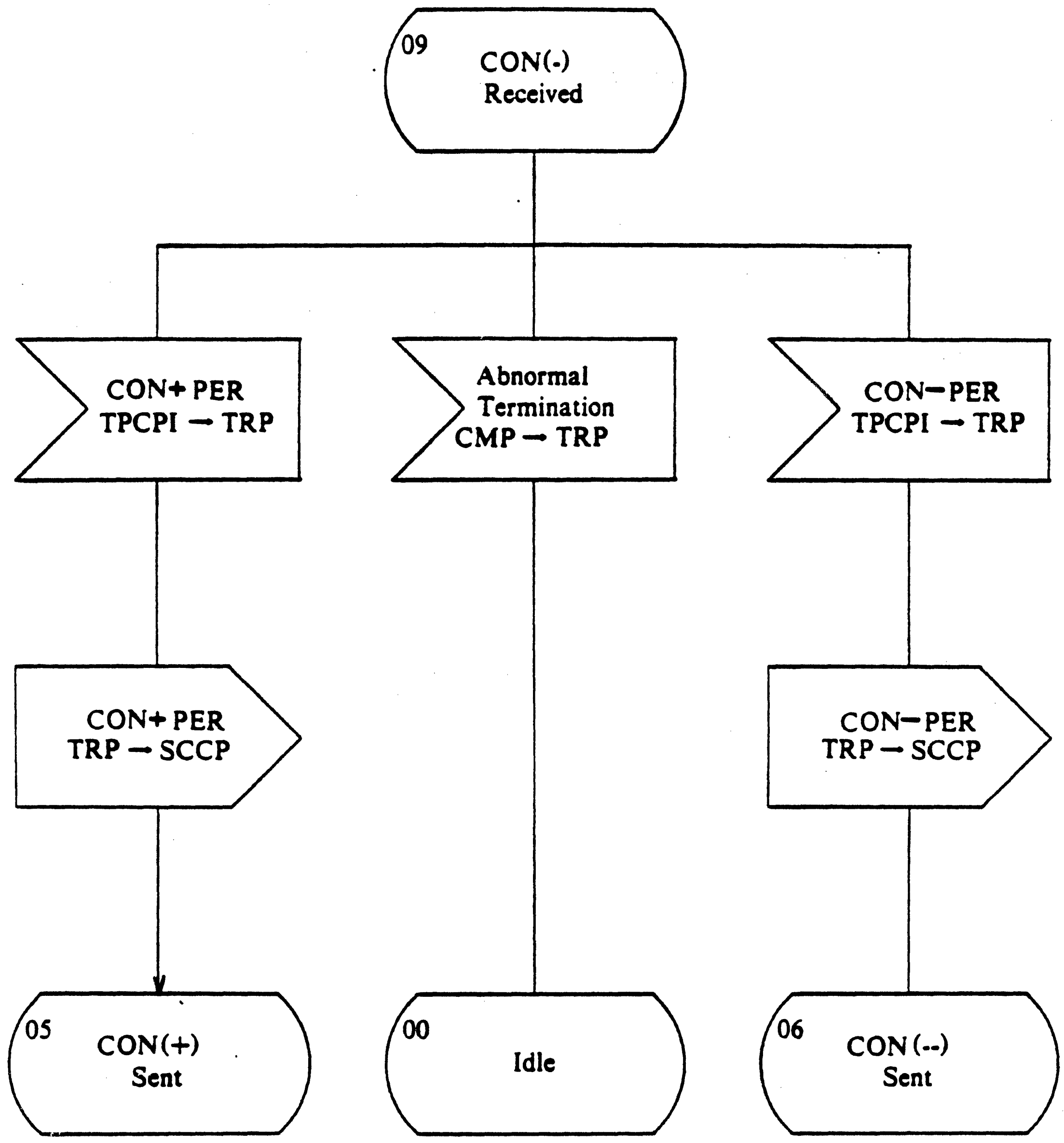


Figure 7/Q.774. Transaction Portion (Sheet 7 of 10)

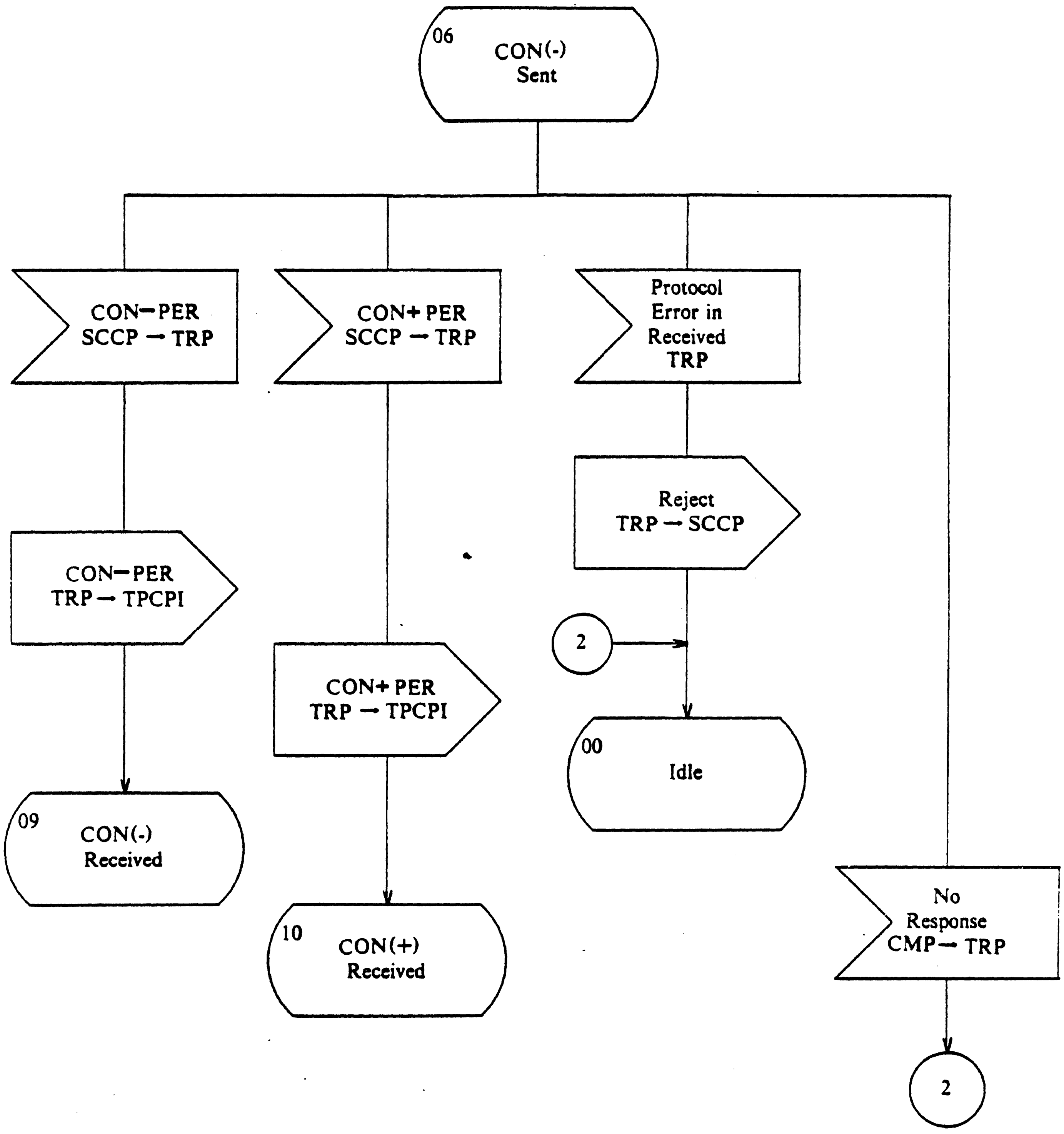


Figure 7/Q.774. Transaction Portion (Sheet 8 of 10)

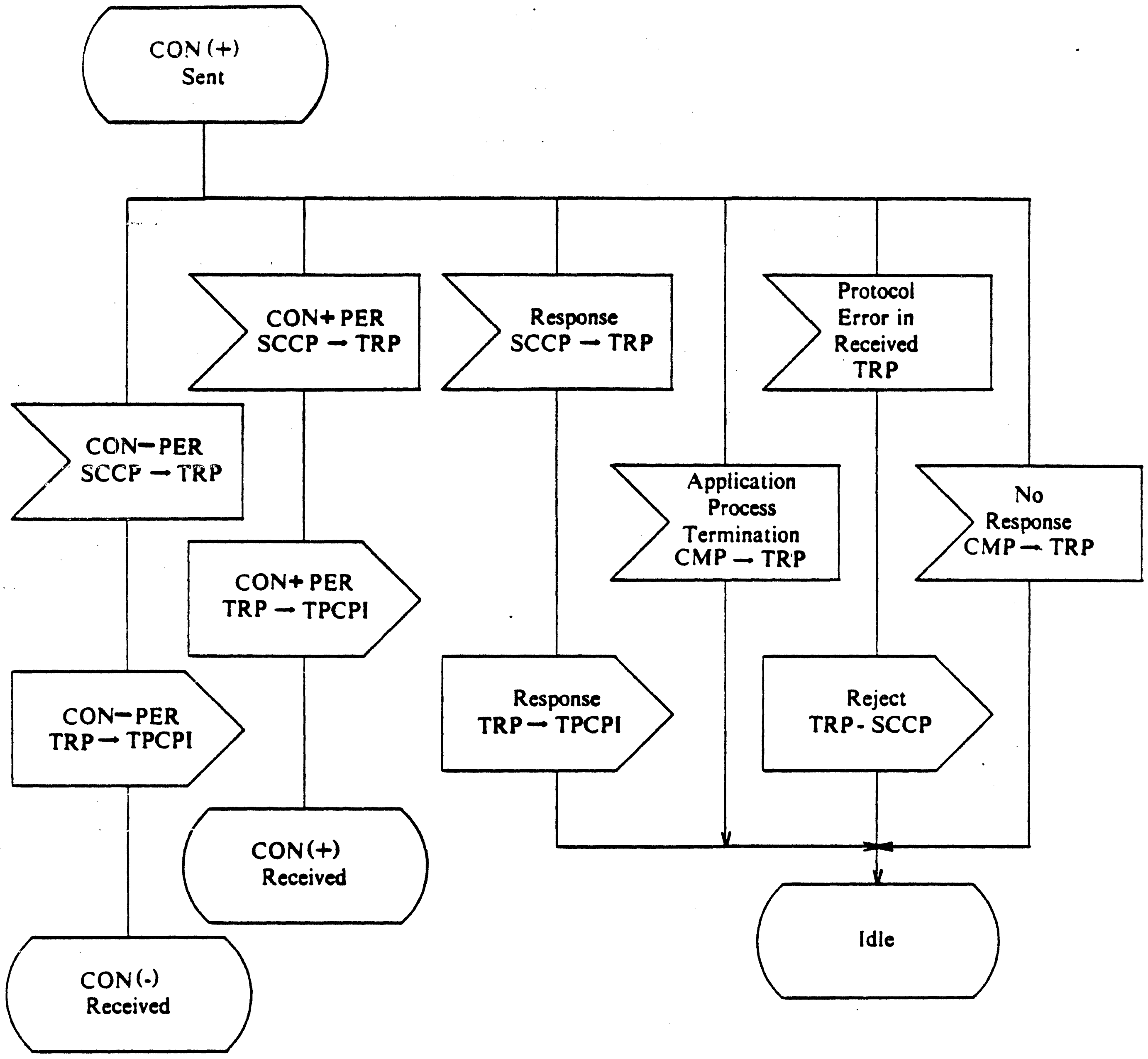


Figure 7/Q.774. Transaction Portion (Sheet 9 of 10) * |

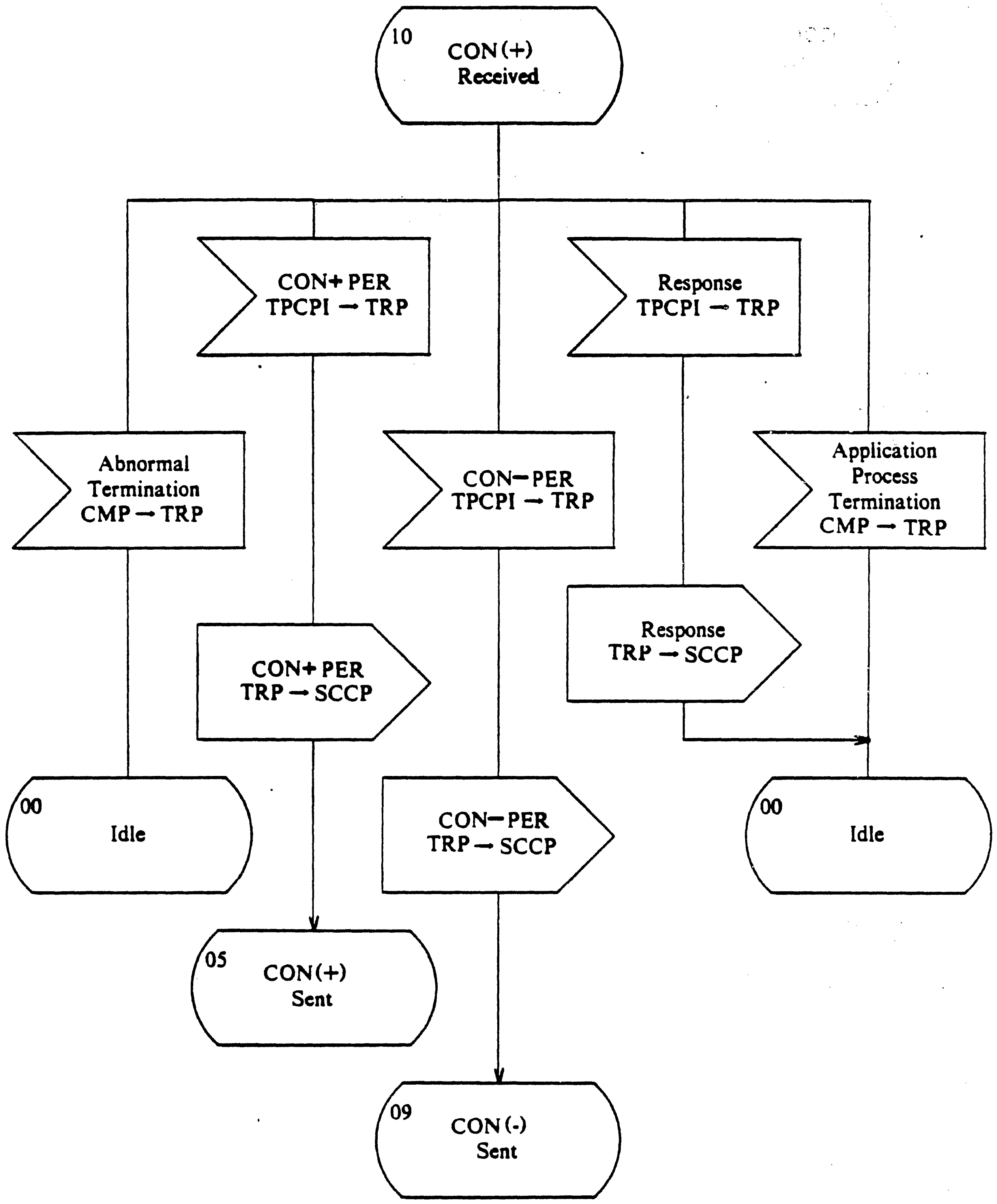


Figure 7/Q.774. Transaction Portion (Sheet 10 of 10) * |

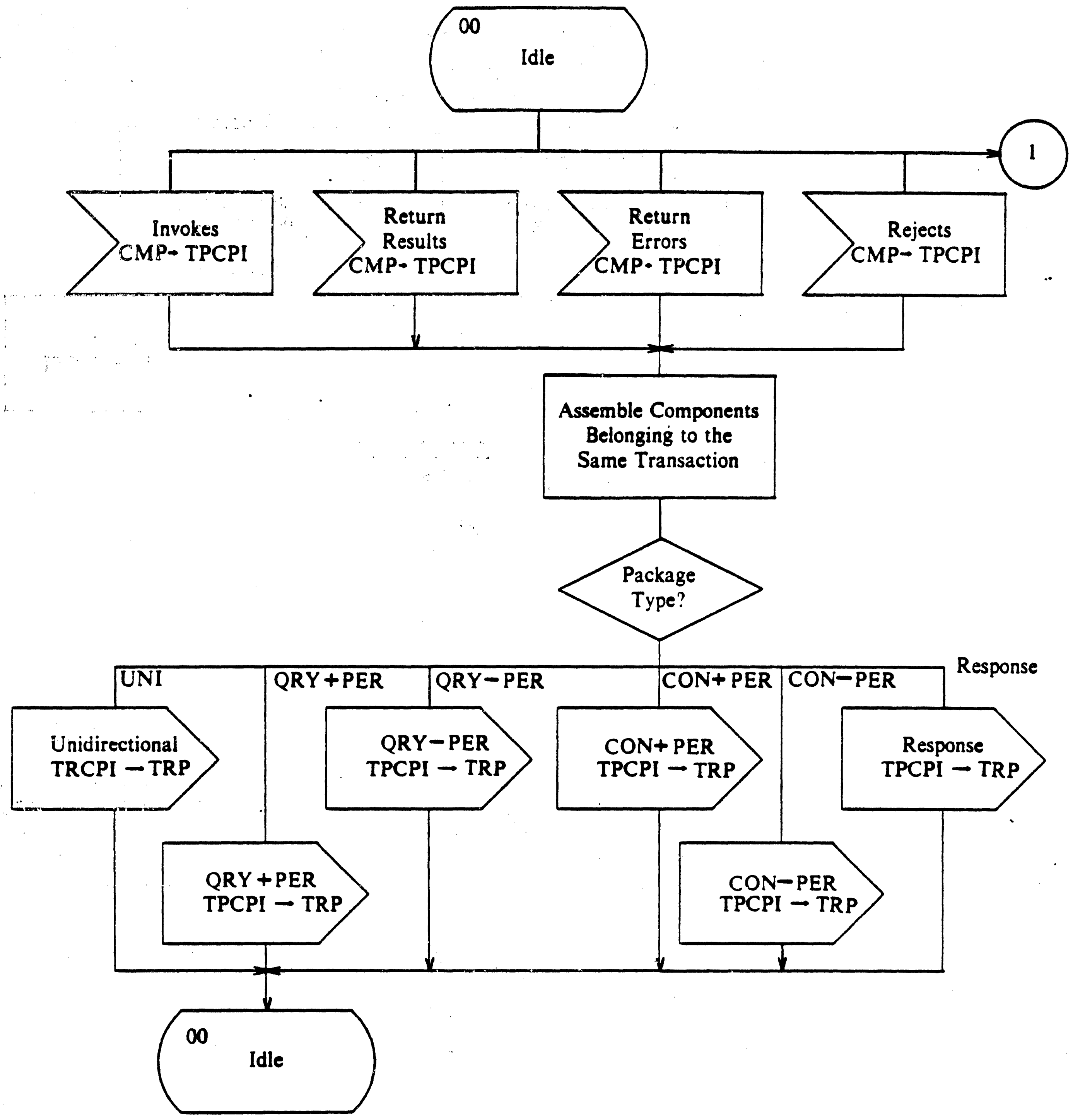


Figure 8/Q.774. Transaction Portion - Component Portion Interface (TPCPI) (Sheet 1 of 2)

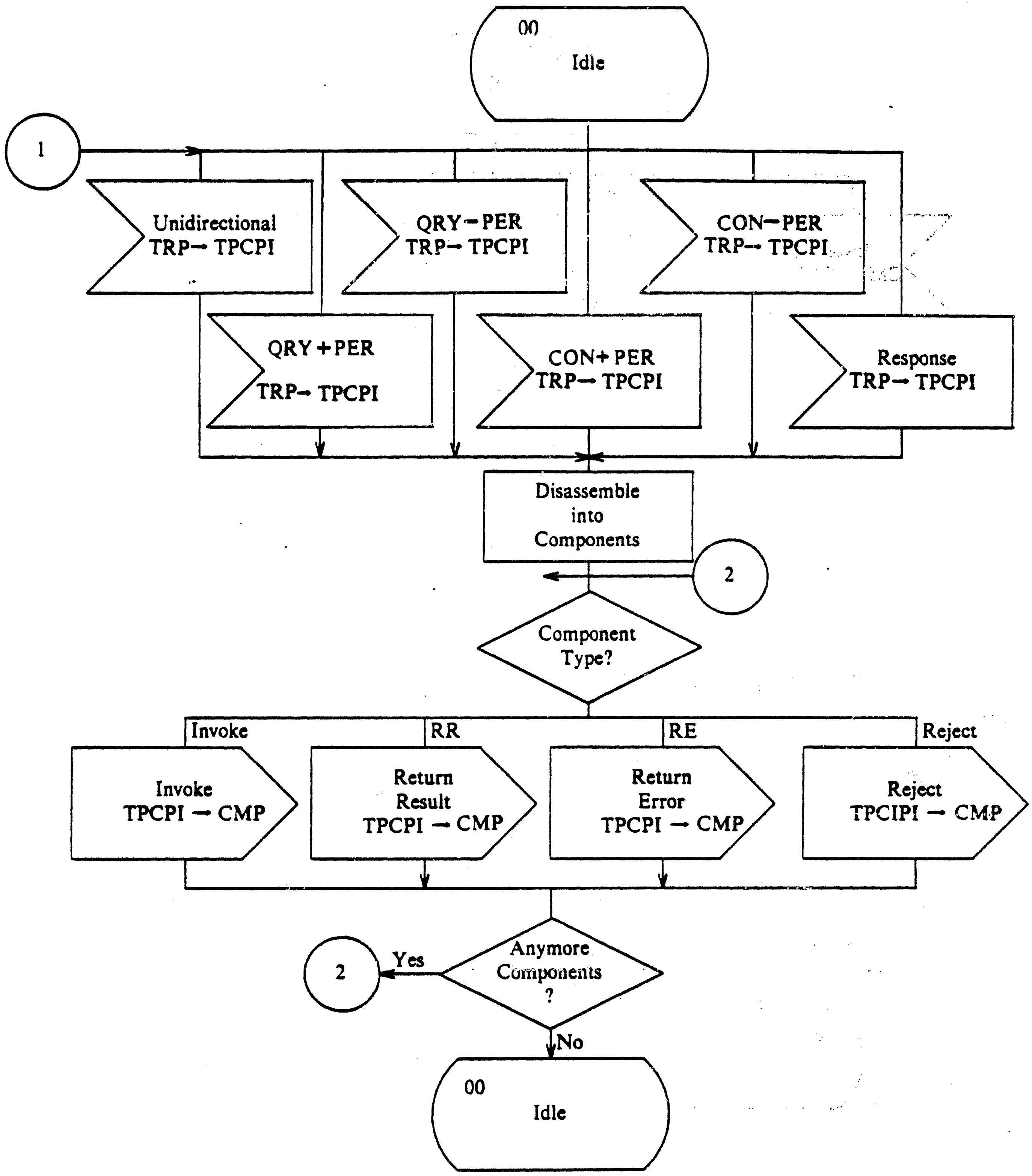


Figure 8/Q.774. Transaction Portion Component Portion Interface (Sheet 2 of 2)

Note: Package Type, Permission Parameters, as well as Transaction and Component IDs will be carried across all functional blocks.

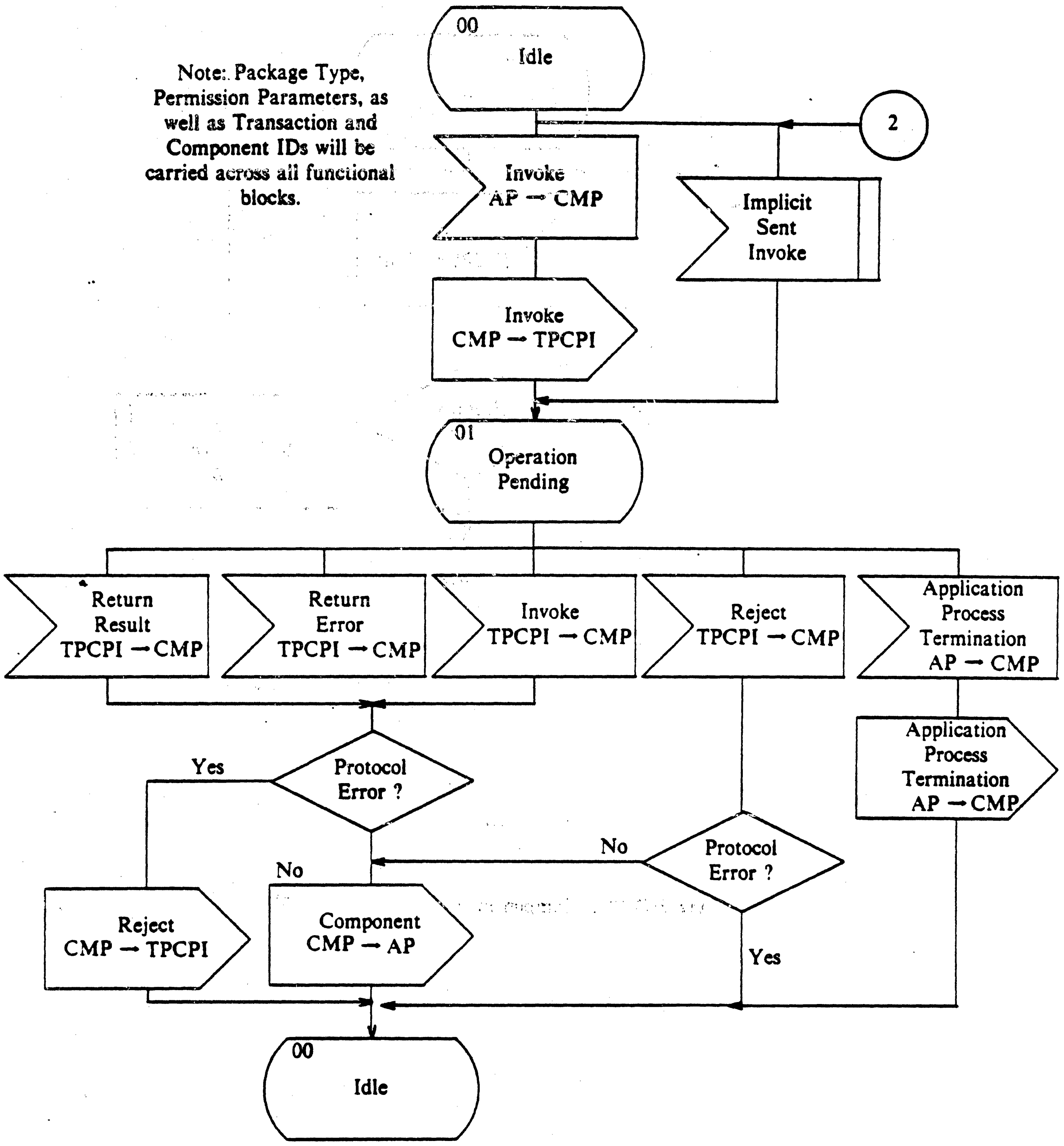


Figure 9/Q.774. Component Portion (Sheet 1 of 3)

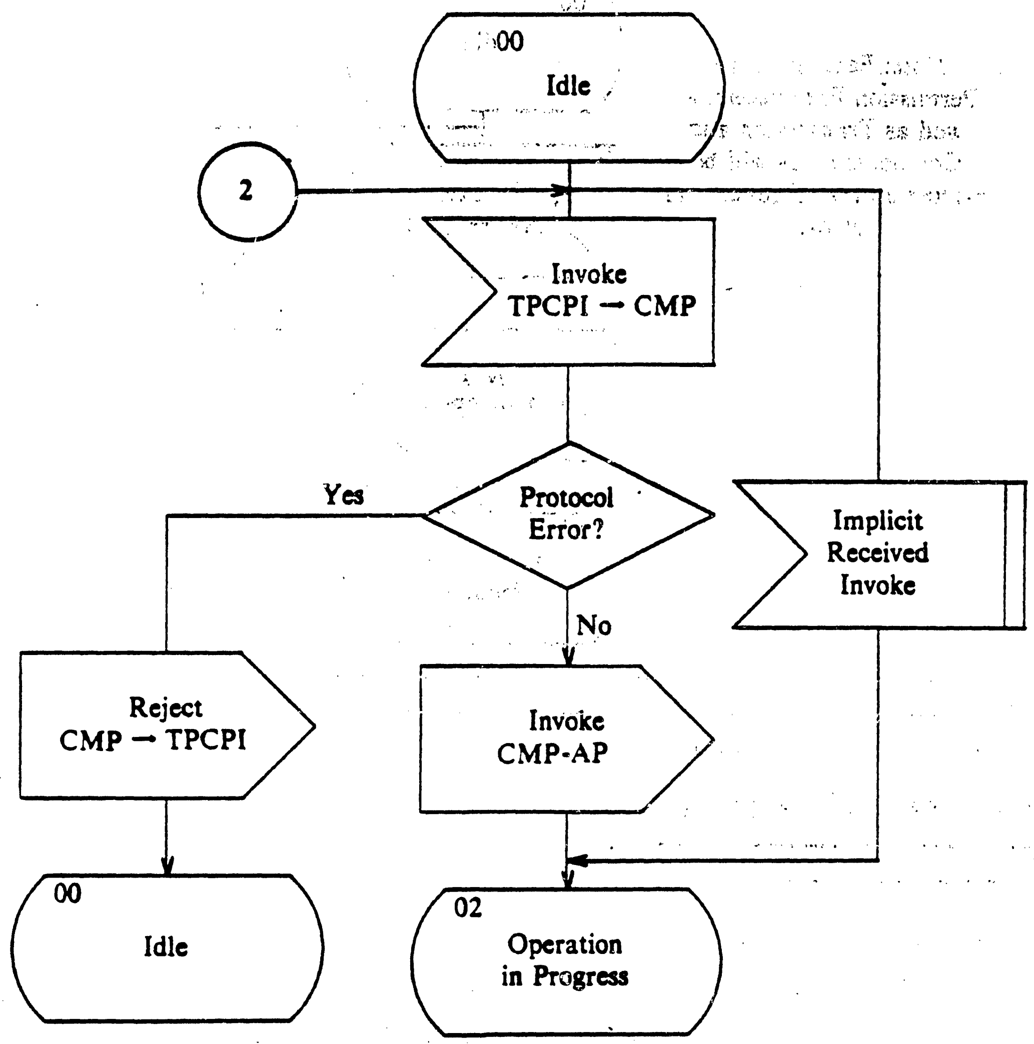


Figure 9/Q.774. Component Portion (Sheet 2 of 3)

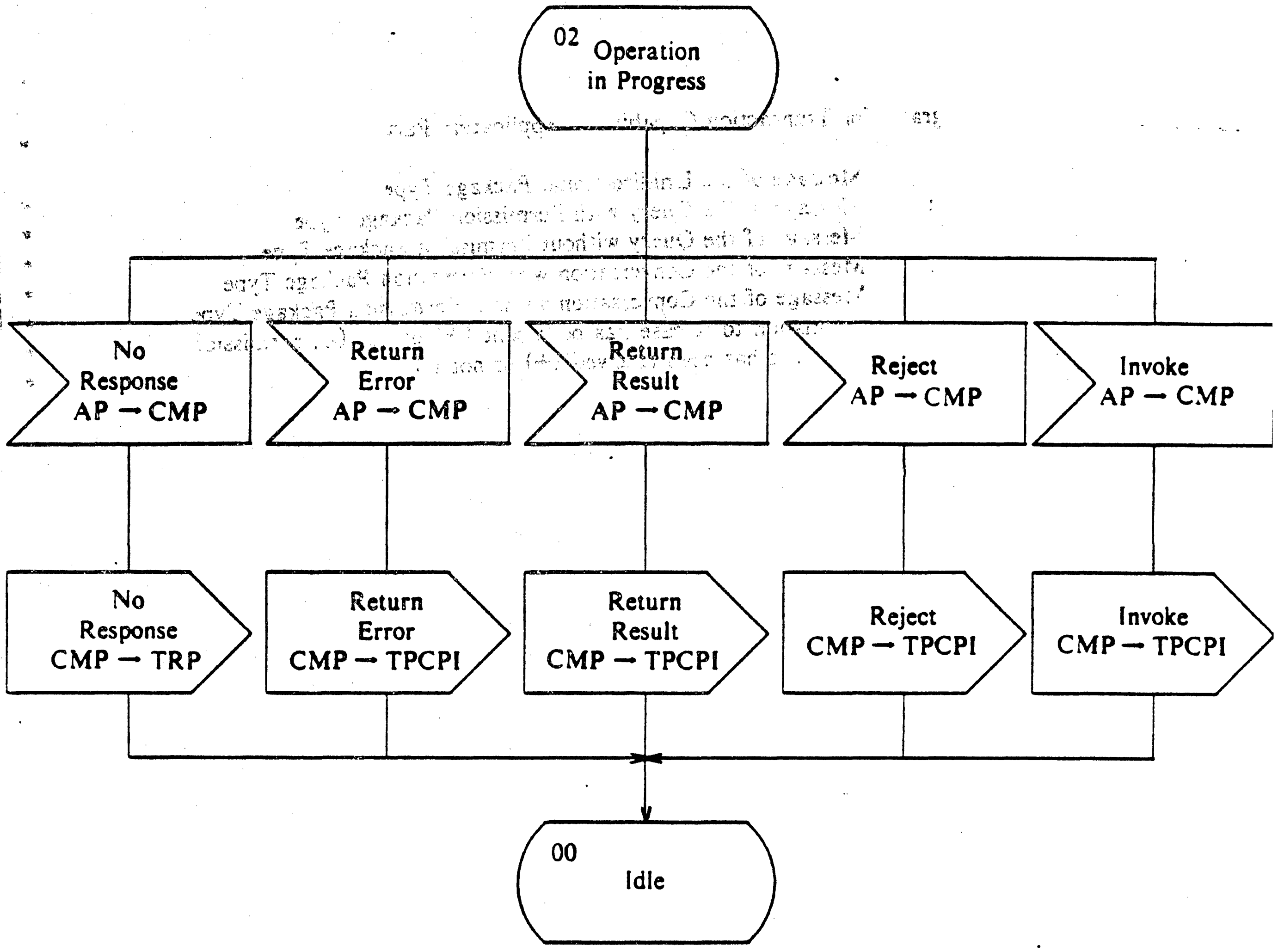


Figure 9/Q.774. Component Portion (Sheet 3 of 3)

APPENDIX I

State Transition Diagrams for Transaction Capabilities Application Part

UNI:	Message of the Unidirectional Package Type
QRY+PER:	Message of the Query with Permission Package Type
QRY−PER:	Message of the Query without Permission Package Type
CON+PER:	Message of the Conversation with Permission Package Type
CON−PER:	Message of the Conversation without Permission Package Type
±,±:	Permission to release has been sent (+) or not (-), permission to release has been received (+) or not (-).