CSDL-AIPS-84-139

ADVANCED INFORMATION PROCESSING SYSTEM (AIPS)

PROOF-OF-CONCEPT SYSTEM

FUNCTIONAL DESIGN

I/O NETWORK SYSTEM SERVICES

March 11th, 1985

Approved: _____
James E. Kernan, Head, I/O Design Team

Approved: _____
Robert N. O'Donnell, Technical Manager

Approved: _____
Philip G. Felleman, Program Manager

The Charles Stark Draper Laboratory, Inc.
Cambridge, Massachusetts 02139

ACKNOWLEDGEMENT

This report was prepared by The Charles Stark Draper Laboratory, Inc. under Contract NAS9-16023 with the Lyndon B. Johnson Space Center of the National Aeronautics & Space Administration.

The major contributors to the design were James E. Kernan, Alton A. Knosp Jr., Gail A. Nagle, and Gary Schwartz. Alan I. Green served as chief reviewer for technical correctness and consistency. Special thanks to Paul Palasek for drawing the data flow diagrams.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION 1

INTRODUCTION AND OVERVIEW

1.1 Introduction

This report contains a description of the functional design of I/O Services for the AIPS Proof of Concept System.

The design methodology employed is based on the use of data flow analysis techniques. The methodology is more fully described in references [6], [7], and [9] .

Section " 2. Data Flow Diagrams" contains the data flow diagrams, which show the functional processes in I/O Services and the data that flows among them. The data flow diagrams are organized in a hierarchical manner. I/O Services is divided into two primary categories: providing services to acquire or deliver data to and from devices such as sensors and effectors, and maintaining the I/O System in functioning order. The former category is referred to as I/O User Communication Services and the latter as I/O Network Management.

Section " 3. Process Descriptions" contains a description of every process. The data flow diagrams, process reference numbers, and identifiers illustrate the hierarchical relationship of the processes.

Section " 4. Data Dictionary" contains a complete list of the data identified on the data flow diagrams and in the process descriptions. A brief description is included with each entry.

Section " 5. Process and Data Location" specifies the physical location of the processes and data. It also identifies whether the process is performed by hardware or by software.

Appendix " A. Data Flow Diagram Symbol Definitions" contains definitions of the symbols used on the data flow diagrams.

Appendix " B. Process Description Format Explanation" contains an explanation of the format of the Process Descriptions.

Appendix " C. Glossary of I/O Terms" contains a glossary of I/O terms used in the AIPS Proof of Concept System.

1.2 Design Overview

The overall design of I/O Network Services separates the I/O Network Manager from I/O User Communication Services so as to support the system level requirements regarding growth, change, and modularity. Specifically, the Network Manager manages the I/O network itself (but not the nonmanagers subscribers and their root interfaces). Each subscriber GPC's I/O User Communication Service manages its own root interfaces to the network. And each User (Application) Function, in conjunction with the I/O User Communication Services, manages its use of the DIUs. This functional separation allows the Network Manager to perform its functions largely independently of the subscribers and without the need to track the hardware or software configurations of the GPC subscribers and the application functions.

The next two sections give a brief overview of the major design characteristics of I/O User Communication Services and I/O Network Management.

1.2.1 I/O User Communication Services

I/O User Communication Services provides two general categories of service. The first covers requests for the transmission and reception of data between a GPC and an I/O device external to the GPC. The second category covers requests for utility services to modify certain characteristics of the first category of service. Transaction selection, bypass clear, and initialization are examples of this second category of service.

The system level performance requirements regarding I/O response time and transport lag significantly influenced the design of I/O User Communication Services.

Since it was desired to provide a user interface that is largely transparent to the actual construction and current connectivity state of the I/O networks, it was necessary to provide a method of mapping a user request for data service into the particulars of networks and DIUs. The design approach was influenced by two significant factors:

- The I/O networks are contention networks, i.e., each GPC subscriber must win a contention before utilizing the network, and the cost of the contention (in time) is not insignificant.

- Many applications require that the time relationship among the reading and writing of data to various sensors and effectors within the same user request be controlled.

These factors led to the design that permits reading and writing to multiple devices on the same network under the umbrella of a single contention sequence: a chain.

It is possible that a user request will involve devices that are connected to different I/O networks. In this case, it is not practical to

provide controlled time relationship across the networks because of the difficulty of acquiring contention control of multiple networks; indeed, any scheme to accomplish this would have to avoid a possible deadlock situation. However, an application system designer can achieve the effects of simultaneous access control over multiple networks by partitioning a single network. This allows a user, for example, to simultaneously read redundant versions of a sensor by placing the redundant versions in different partitions of the network.

Since most application functions execute the same I/O transactions repeatedly, the design utilizes the concept of predefined chains that are for the most part fixed in nature. A design that permitted only totally rigid chains would, however, lead in some instances to the need for a large number of chains, some being slight variations of each other. To avoid this, the Transaction Selection feature is provided. This service permits the user to specify which of the transactions within a chain will be performed and which will not be performed. A selection, once made, is effective for all occurrences of the chain until a new selection is made.

To support the need for a general initialization feature at the initiation of an application function, the Function Initialization feature is provided. Its invocation restores all of an application function's data transmission and reception requests to their initial state.

The consideration of I/O communication errors and their causes influenced the design in a number of ways.

A GPC, a DIU, or the network itself can be the source of an I/O error. The decision to functionally separate the I/O Network Manager from I/O User Communication Services places the responsibility for the isolation of network and babbling subscriber faults with the Network Manager and the responsibility for isolating GPC interface and DIU faults with the I/O User Communication Services.

The approach used to isolate DIU faults is to successively omit errored transactions from the chain until it no longer accesses the suspect DIU: this action is referred to as transaction bypass. However, since network errors can result in the same effects, it is necessary for the network manager to inform each of the GPC subscribers on the network following a repair operation so that any bypassed transactions can be reinstated; this operation is called Bypass Clear . In general, when a transaction within a chain is bypassed, the chain as a whole is shortened by the duration of that transaction. However, in the case of a partitioned network, an equivalent time pad is substituted for the bypassed transaction so as to maintain the simultaneity of operations on the various parallel partitions.

To insure nonconflicting usage of a particular I/O network by the same or multiple users within the same GPC, the design approach is to perform all requests of either the data or utility type serially and without overlap. Thus, for example, a Transaction Selection request that is entered for a particular chain while that chain is being performed for some other request will not be executed until processing for the current

chain is completed. However, the design places no restriction on the simultaneous performance of operations on separate I/O networks.

1.2.2 I/O Network Management

An I/O network is a reconfigurable, virtual bus which allows GPC subscribers to access input/output devices or DIUs connected to the bus. The reconfigurability feature is allowed by the 5-ported nodes which join the various communications elements into a network [5]. These nodes provide more than the minimum number of links required to form the bus. Under control of the I/O Network Manager, the spare links can be brought into service in response to a network failure, thus restoring I/O service and increasing the reliability of the network. Furthermore, on-line modifications of the network are permitted. These modifications can range from the reinstallation of a repaired node to the addition of new nodes and links.

A GPC subscriber to the network may be a Fault Tolerant Processor (FTP) consisting of two or three synchronously operating, simplex channels. Each channel can have a connection to one and only one node of a particular network. This root node is connected to its channel by a root link. Thus a triplex GPC may have up to three root links to a particular I/O network. However, it may have two or only one root link. A simplex GPC can have only one root link to a network. Regardless of the level of redundancy of each GPC, the data on the network is simplex data from a single source. The replication of congruent data for each channel is handled by the FTP interchannel data exchange mechanism [4].

A Network Manager is a process operating within one of the GPC subscribers. The primary function of this process is to maintain the health of the network by finding faulty components and reconfiguring the active network to exclude them. To do this it uses a data base which reflects the actual physical connections in the network and real time data which it periodically collects from the network nodes themselves.

Networks may serve the I/O needs of several GPCs (a regional network) or only one (a local network). In the case where several GPC utilize a network, only one of them (at any one time) will host the Network Manager of that network. The choice of this host processor should reflect the consideration that the greater the number of root links a GPC has to a network, the greater its ability to maintain a connection to the network.

If a network is dedicated to only one GPC, a unique network configuration is possible, namely a partitioned network. Such a network may be partitioned into as many subnetworks as the GPC owner of the network has root links. These subnetworks are a set of redundant parallel virtual buses, each conducting I/O operations with redundant, parallel DIUs. Within each subnetwork are a certain number of spare links which allow failures to be repaired intrapartition. An advantageous feature of this configuration is that while such a repair is taking place, other parti-

tions can operate normally allowing I/O functions to continue uninterrupted. Nevertheless, the potential to merge two or more partitions is available in the event that intrapartition repair becomes impossible. For example, in the event of a root link failure, I/O devices in the temporarily isolated partition can be brought back on the bus by utilizing one of the spare links that connect two partitions of the network. If critical information is at stake, such a departitioning could be used to attempt recovery of even one isolated node.

While a GPC may have several root links to an unpartitioned network, only one of these may be active at any one time. Similarly, in a partitioned network, each partition may have only one active root link to its GPC. This is to prevent two channels from simultaneously utilizing the bus and corrupting each other's signals. The manager of a network does not in general control the root link connection of a GPC to a network. This must clearly be the case when a GPC accesses a network but does not manage it. In such a distributed system, the manager may not have sufficient data to control the network interface to its nonhost GPCs. Thus, the manager will configure each root node so that the port attached to a root link is always active, i.e. capable of transmitting and receiving data. The GPC root link selection process will then choose which root node it will actively communicate through. The manager process will control the configuration of its host GPC's interface only when it is growing the network and when it is reconfiguring the network in response to a network failure.

The manager of the network can isolate network faults and restore I/O service in the face of several types of network faults. These include a passively failed node or node port, a babbling node or subscriber, and a node which responds out of turn when other nodes are addressed. The manager determines the identity of the faulty element and reconfigures the network so as to isolate that element. The manager periodically tests failed nodes and node ports to determine if the failure was transient in nature.

Finally, the manager of an I/O network performs its functions such that they are largely transparent to all applications using the network for I/O service.

SECTION 2

DATA FLOW DIAGRAMS

The top level organization of I/O Services functions is depicted in Figure 2-1. As illustrated, the functions are divided into two categories: I/O User Communication Services processing and I/O Network Manager processing. I/O User Communication Services data flow diagrams are contained in section "2.1 I/O User Communication Services" on page 2-7. I/O Network Manager data flow diagrams are in section "2.2 I/O Network Manager" on page 2-31.

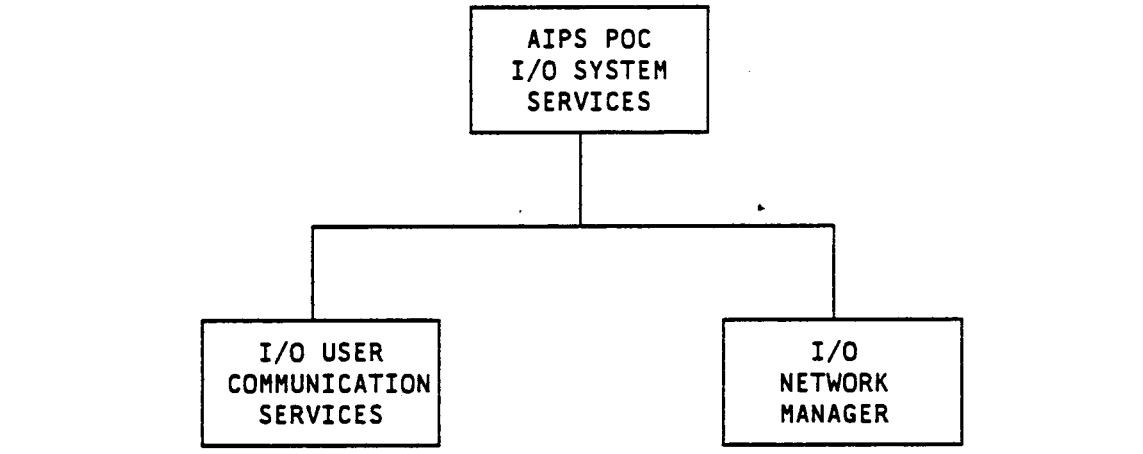


Figure 2-1. AIPS Proof of Concept I/O System Services - Top Level

The context of the I/O System Services data flow within the AIPS Proof of Concept System is shown in Figure 2-2. The data flow between the major categories of I/O System Services is shown in Figure 2-3.

This page is intentionally blank.

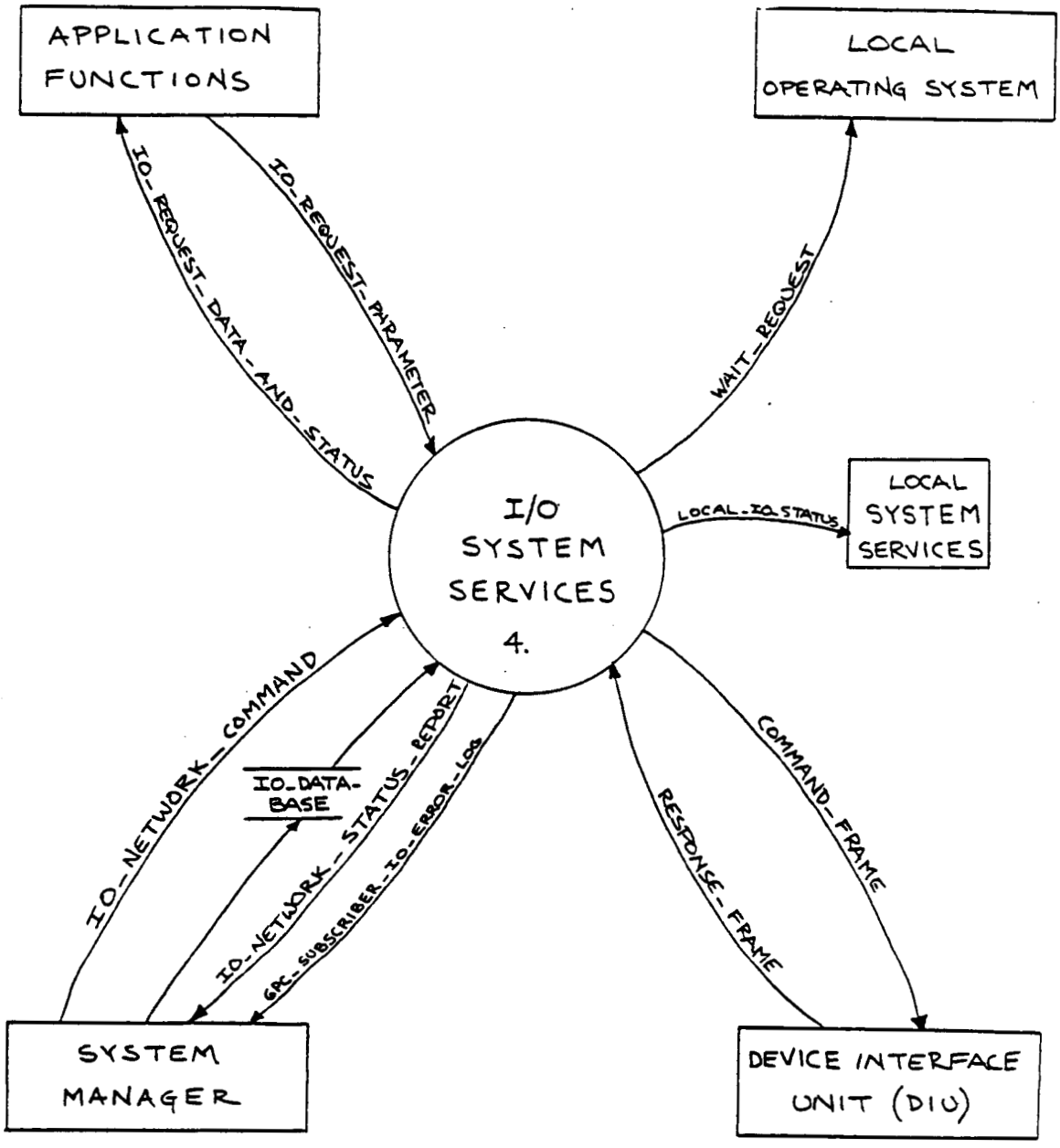


Figure 2-2. Context of I/O System Services

This page is intentionally blank.

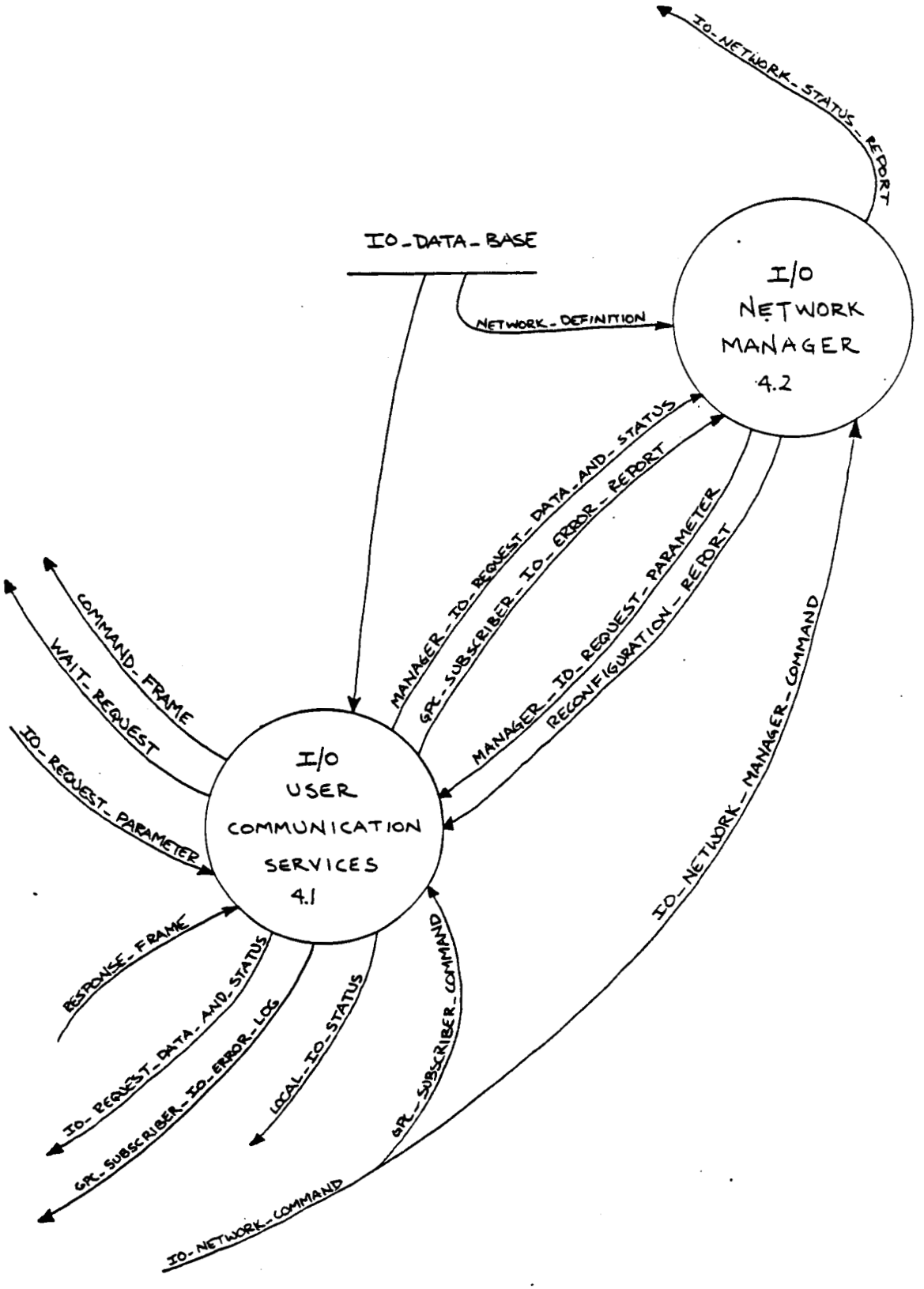


Figure 2-3. I/O System Services - 4.

This page is intentionally blank.

2.1 I/O User Communication Services

I/O User Communication Services provides two general categories of service. The first covers requests for the transmittal and retrieval of data between a GPC and some I/O device external to the GPC. The second category covers requests for utility services to modify certain characteristics of the first category of service. Transaction selection, bypass clear, and initialization are examples of this second category of service. All of these services are processed by I/O Request Processing. I/O User Communication Services data flow diagrams are in Figure 2-5 on page 2-11 through Figure 2-14 on page 2-29 . The hierarchical relationship of these flow diagrams and processes is shown in Figure 2-4.

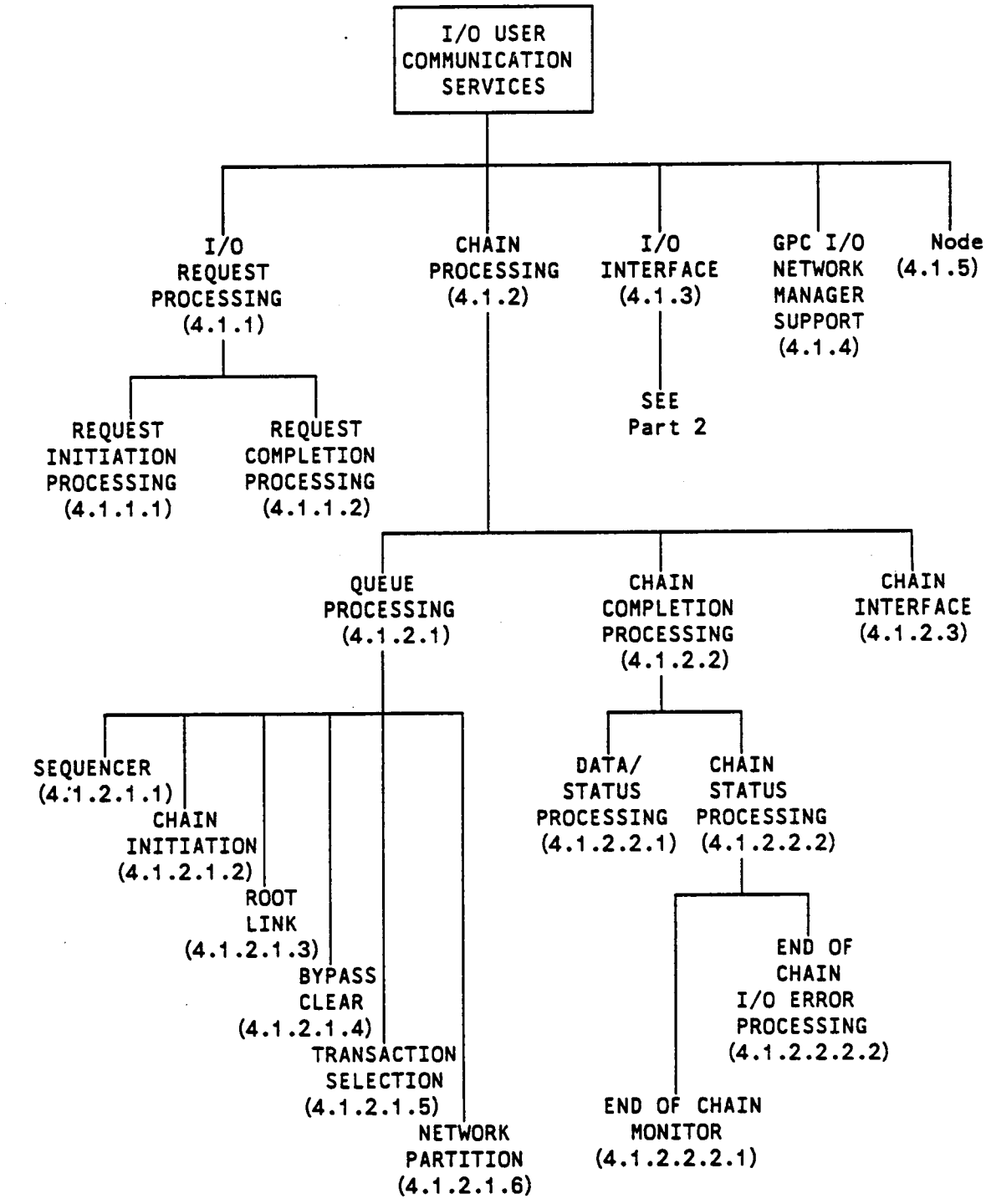


Figure 2-4. I/O User Communication Services (Part 1 of 2): Data Flow Diagram Processing Hierarchy

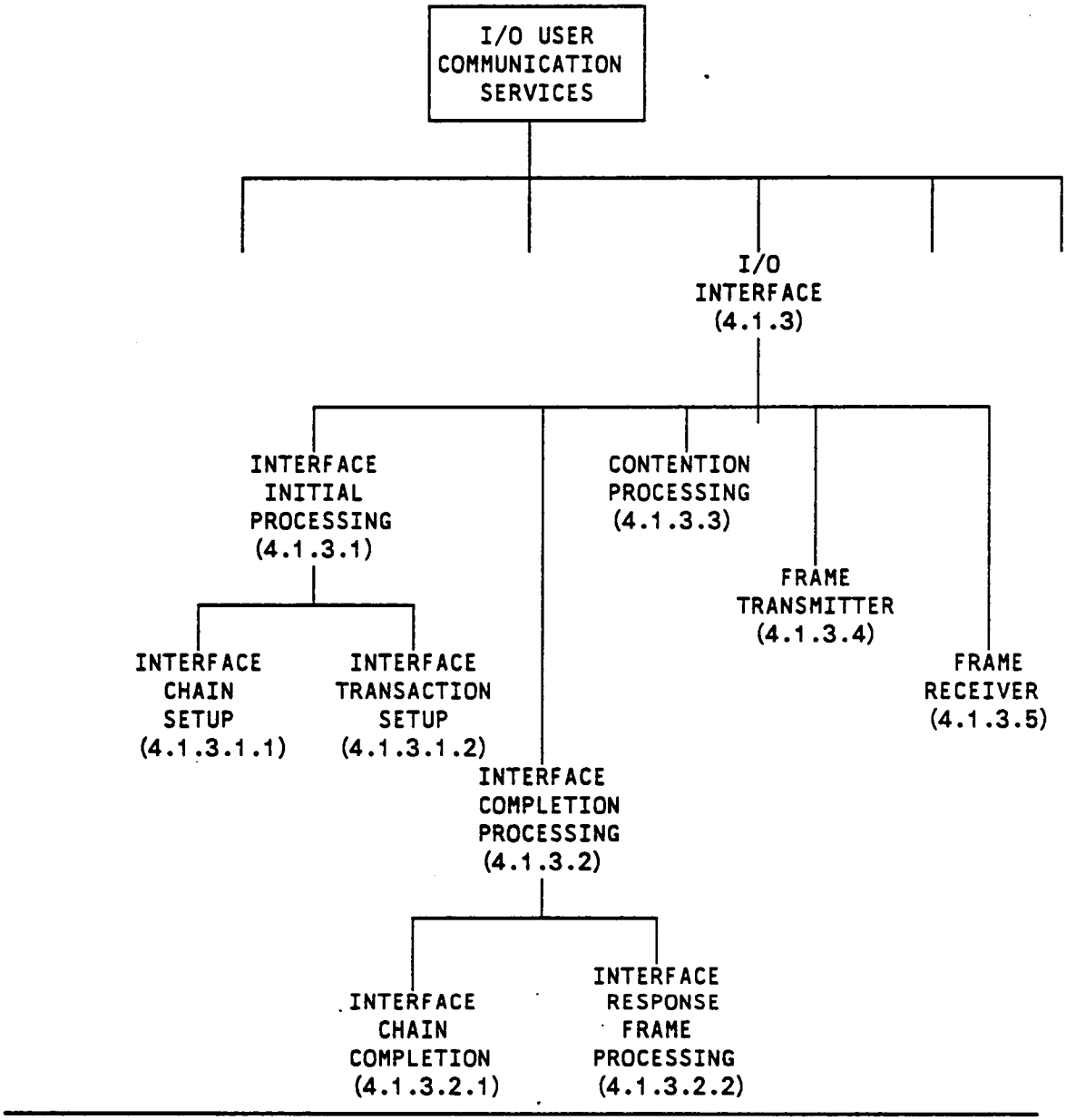


Figure 2-4. I/O User Communication Services (Part 2 of 2): Data Flow Diagram Processing Hierarchy

This page is intentionally blank.

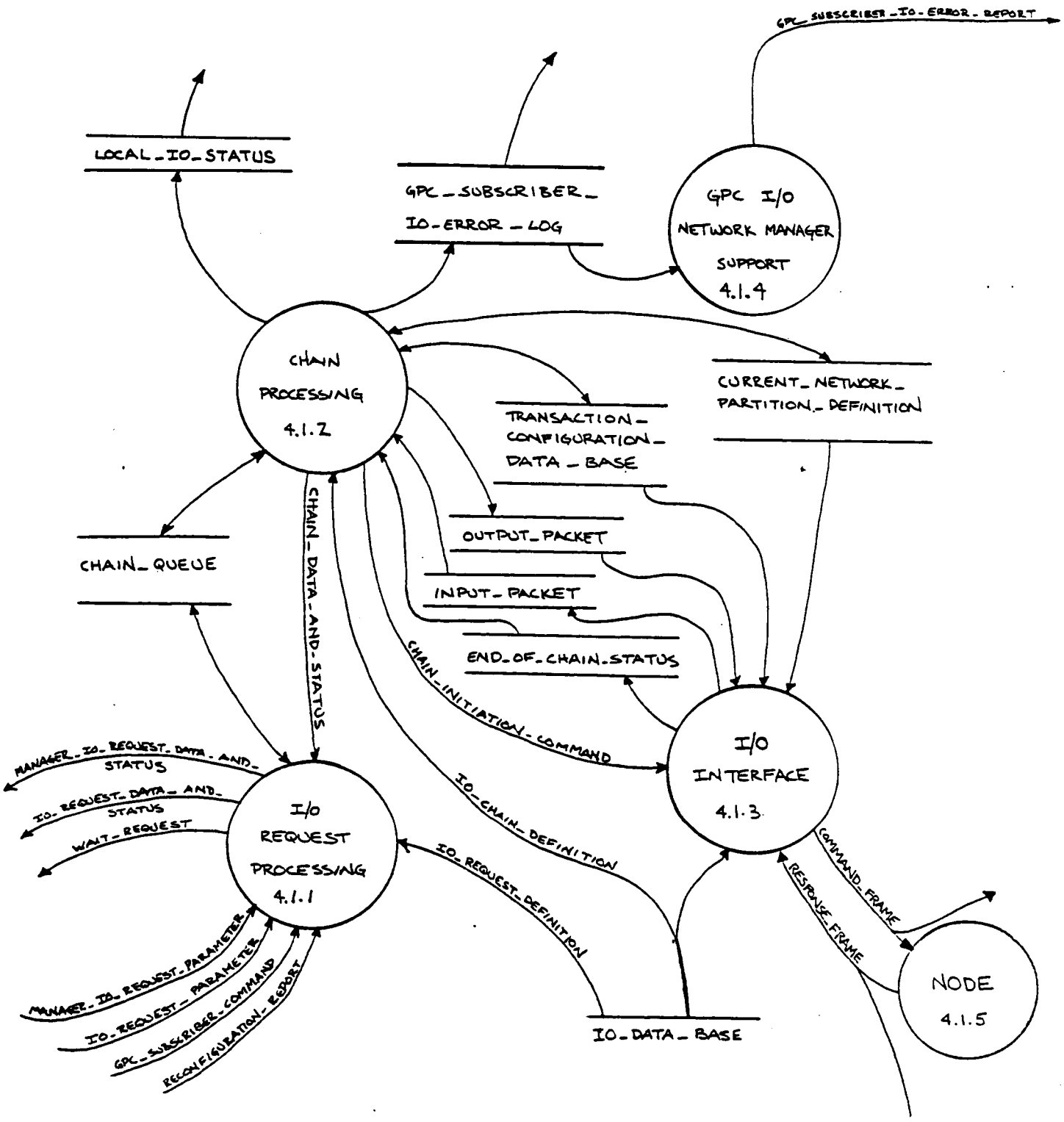
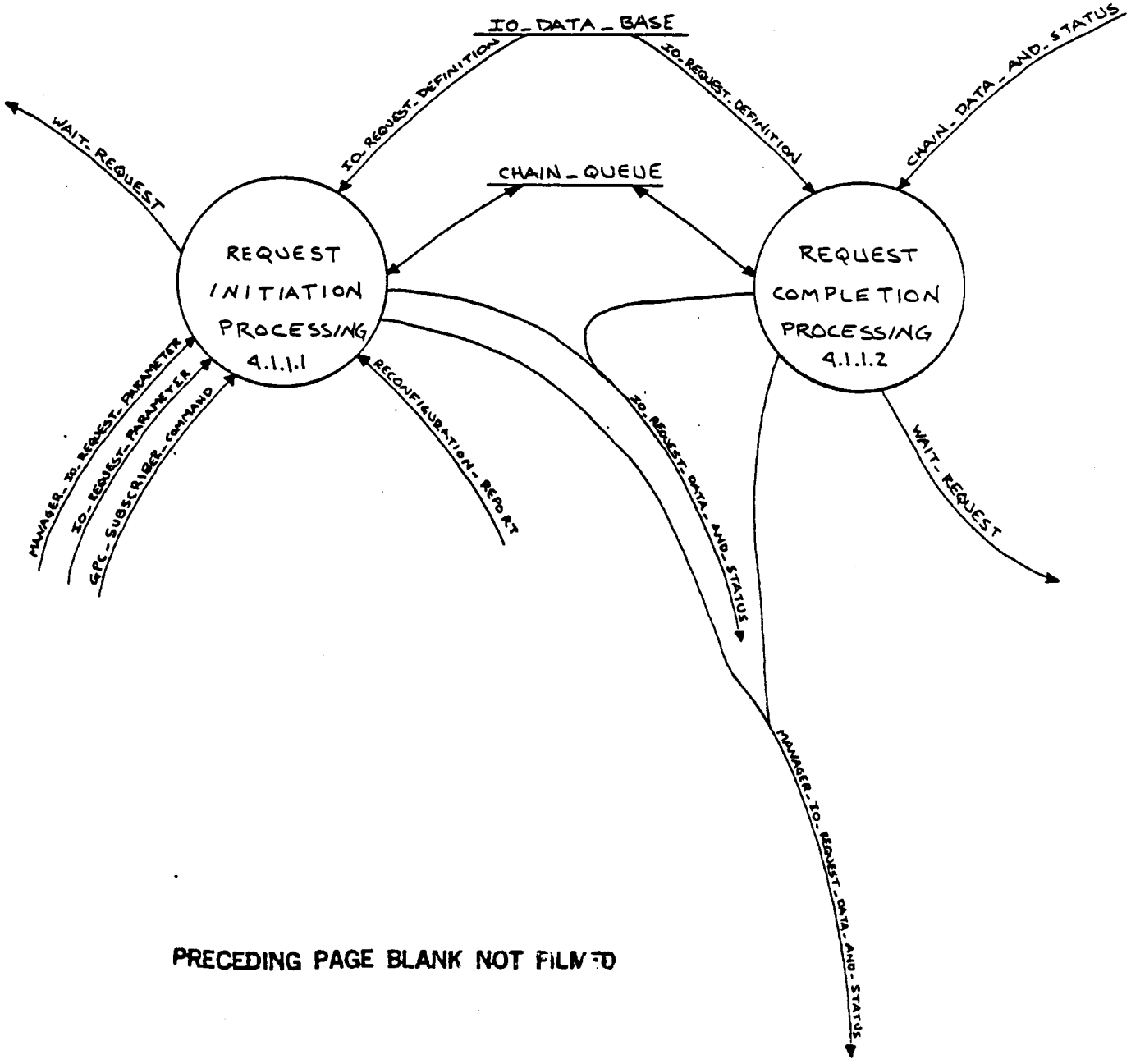


Figure 2-5. I/O User Communication Services - 4.1

This page is intentionally blank.



PRECEDING PAGE BLANK NOT FILMED

Figure 2-6. I/O Request Processing - 4.1.1

This page is intentionally blank.

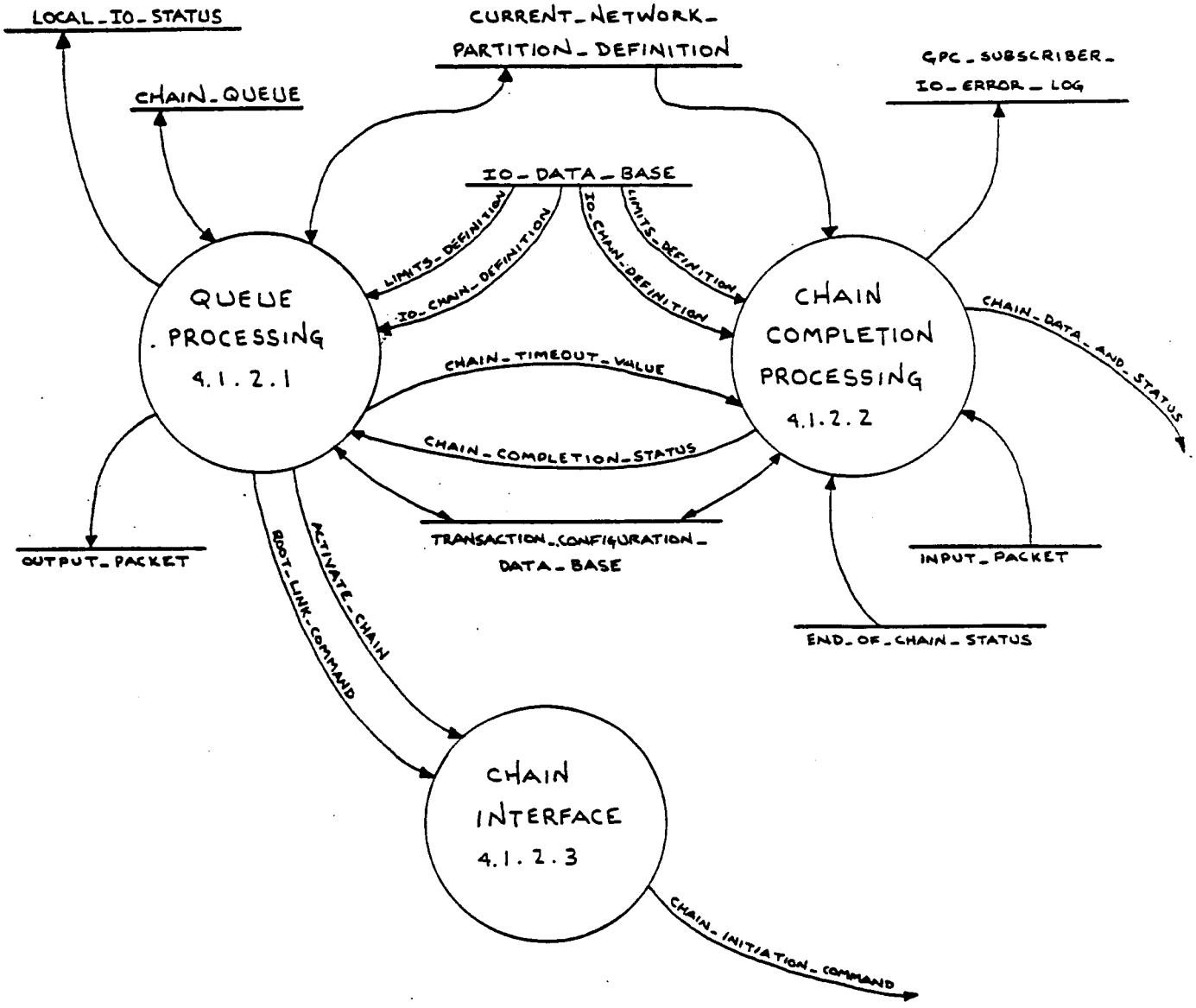


Figure 2-7. Chain Processing - 4.1.2

This page is intentionally blank.

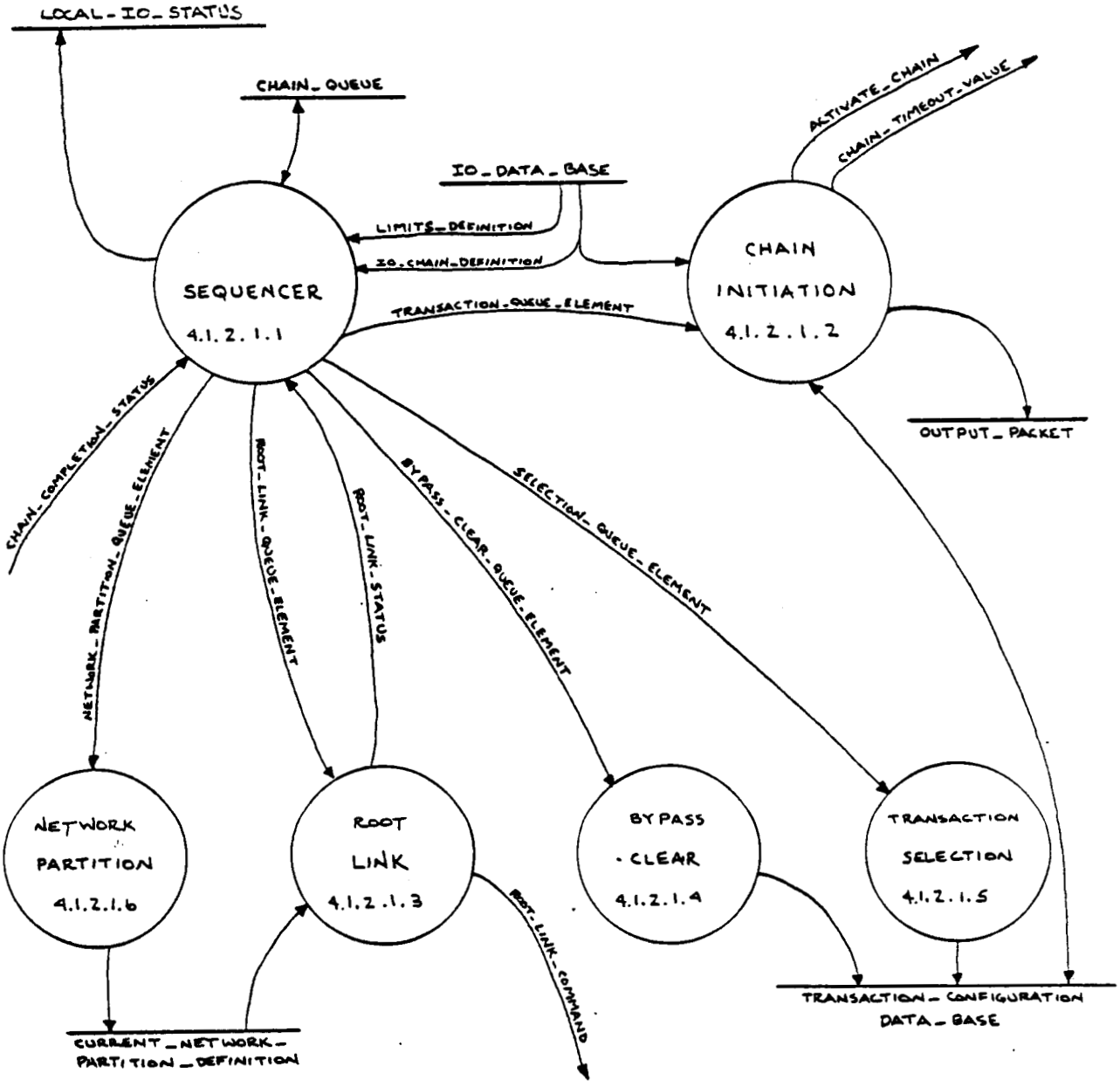


Figure 2-8. Queue Processing - 4.1.2.1

This page is intentionally blank.

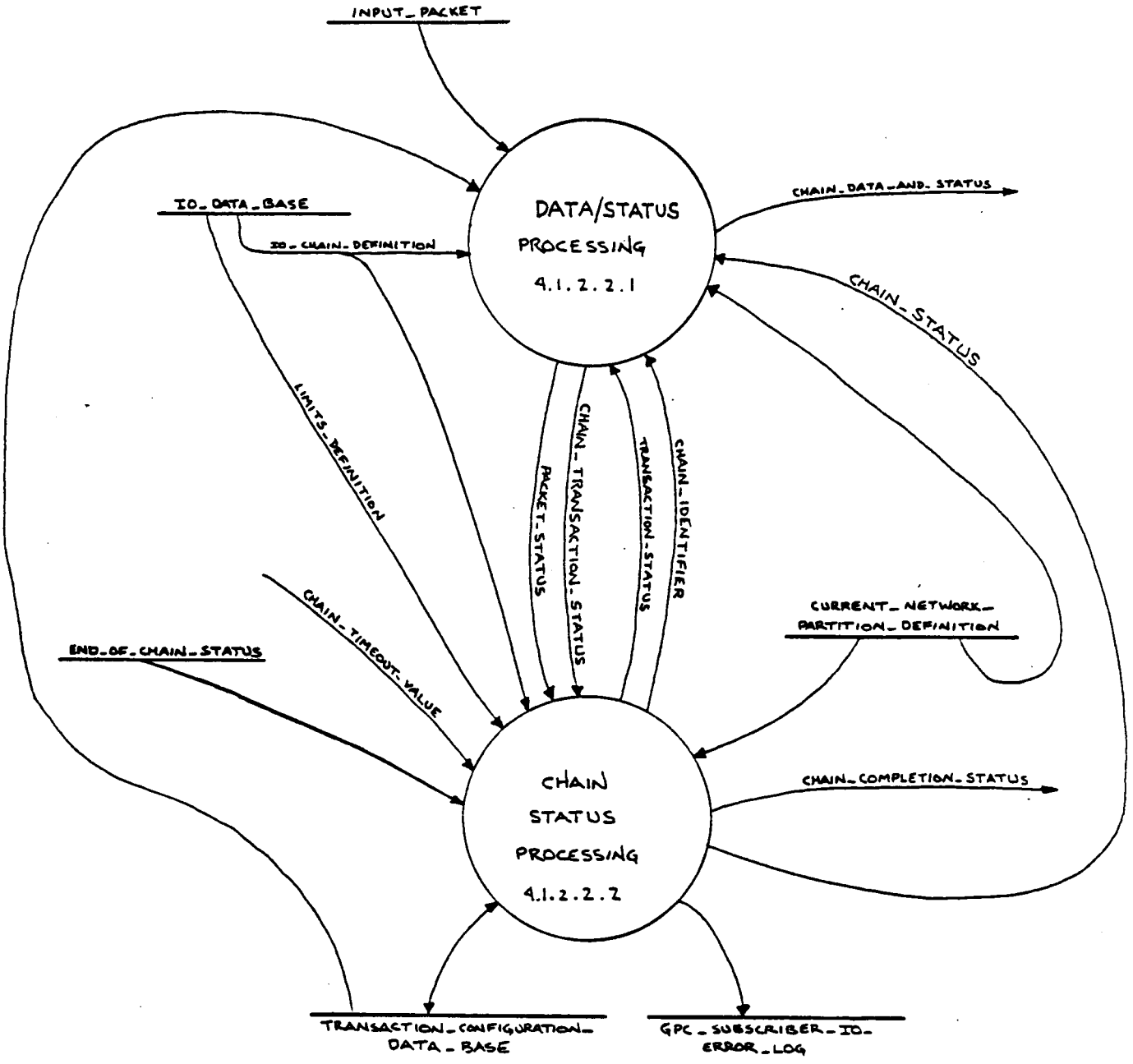


Figure 2-9. Chain Completion Processing - 4.1.2.2

This page is intentionally blank.

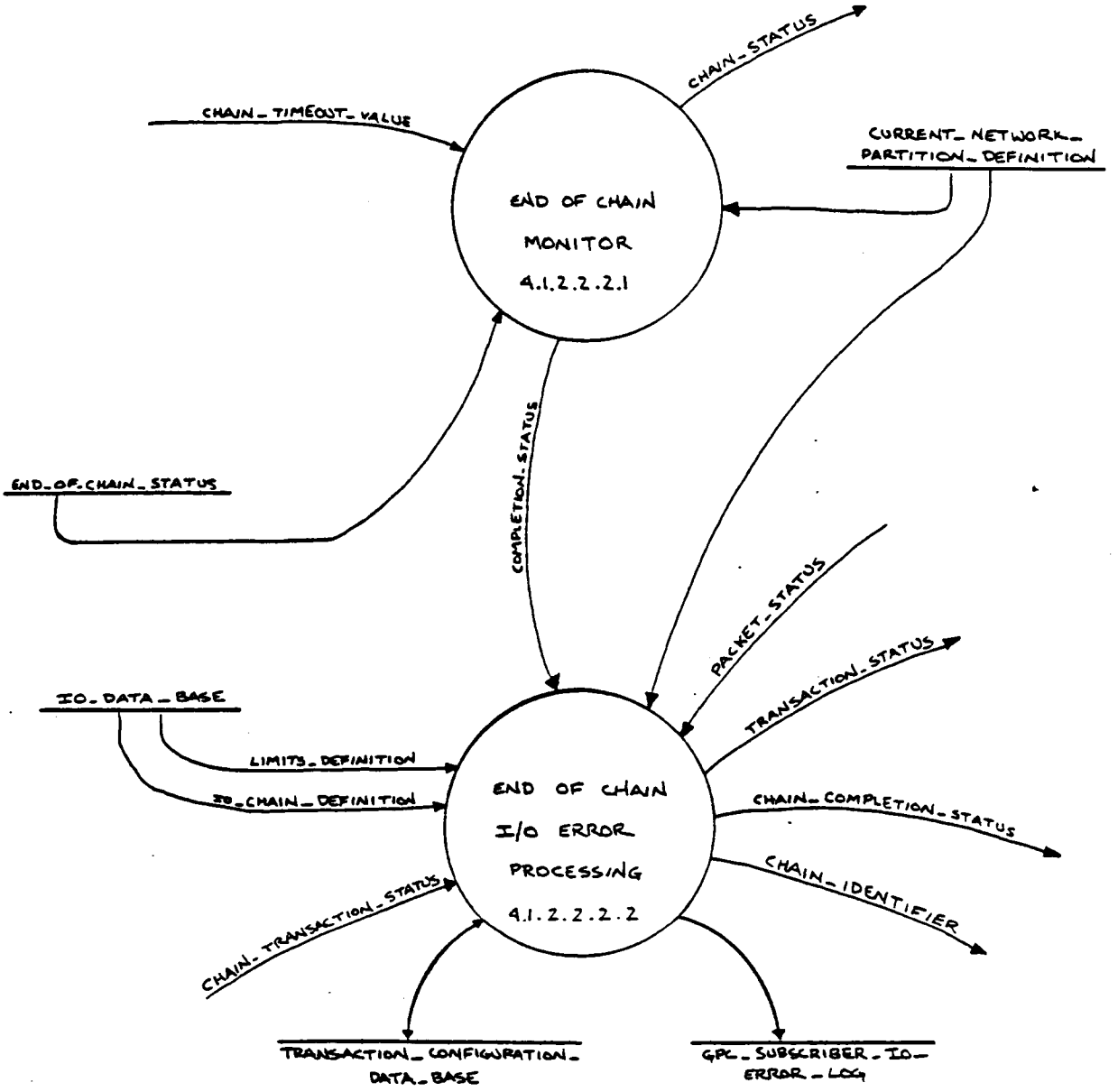


Figure 2-10. Chain Status Processing - 4.1.2.2.2

This page is intentionally blank.

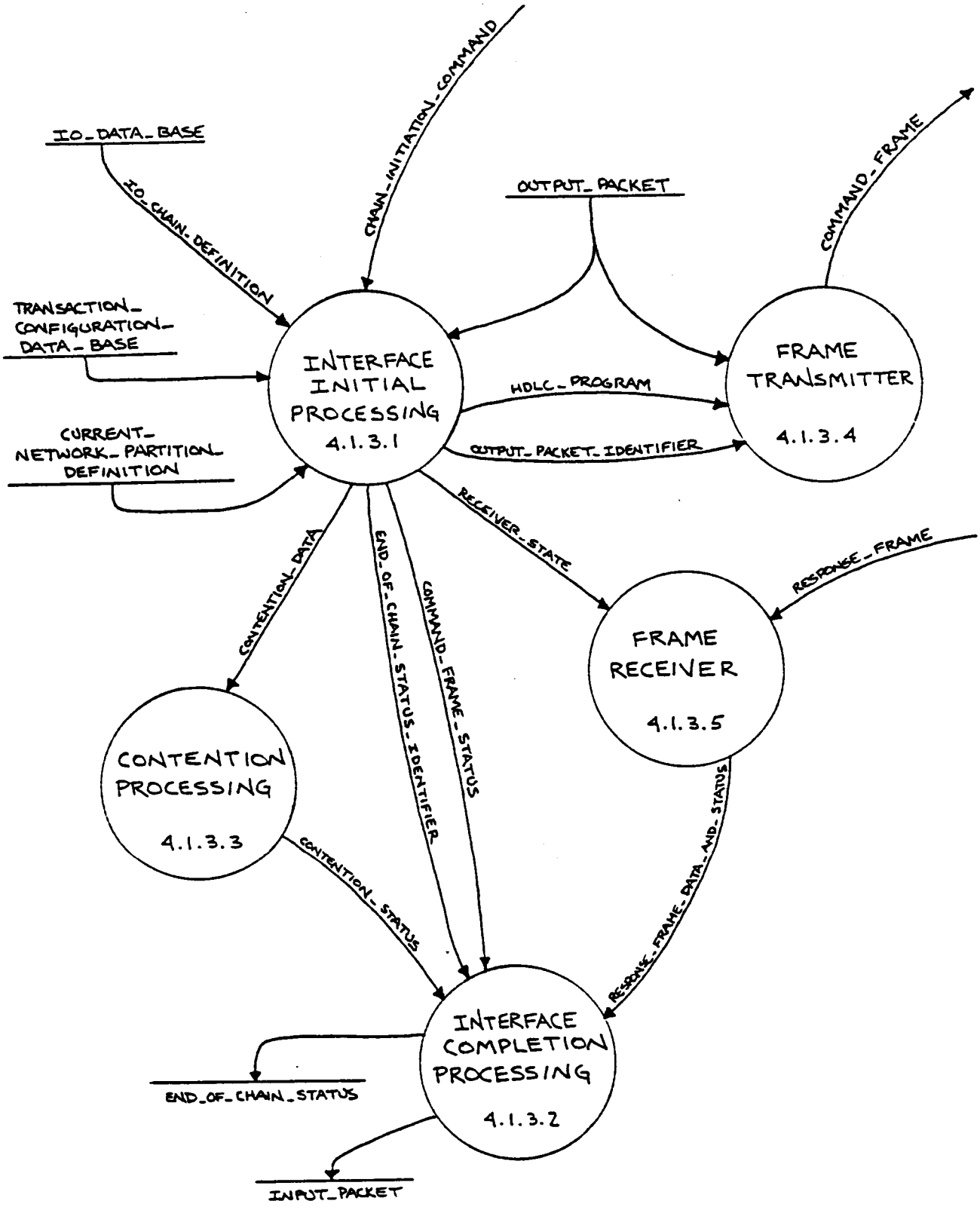


Figure 2-11. I/O Interface - 4.1.3

This page is intentionally blank.

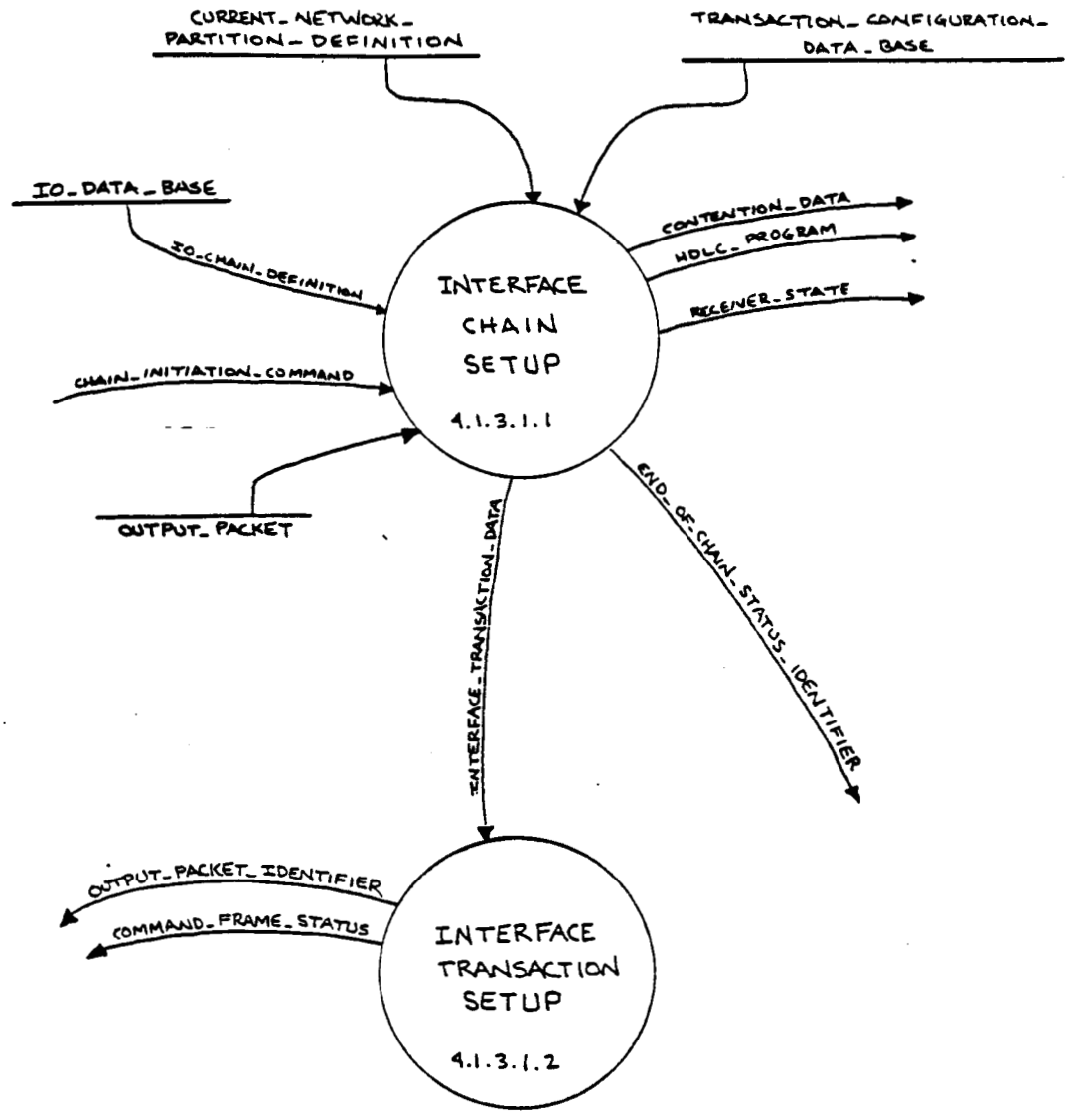
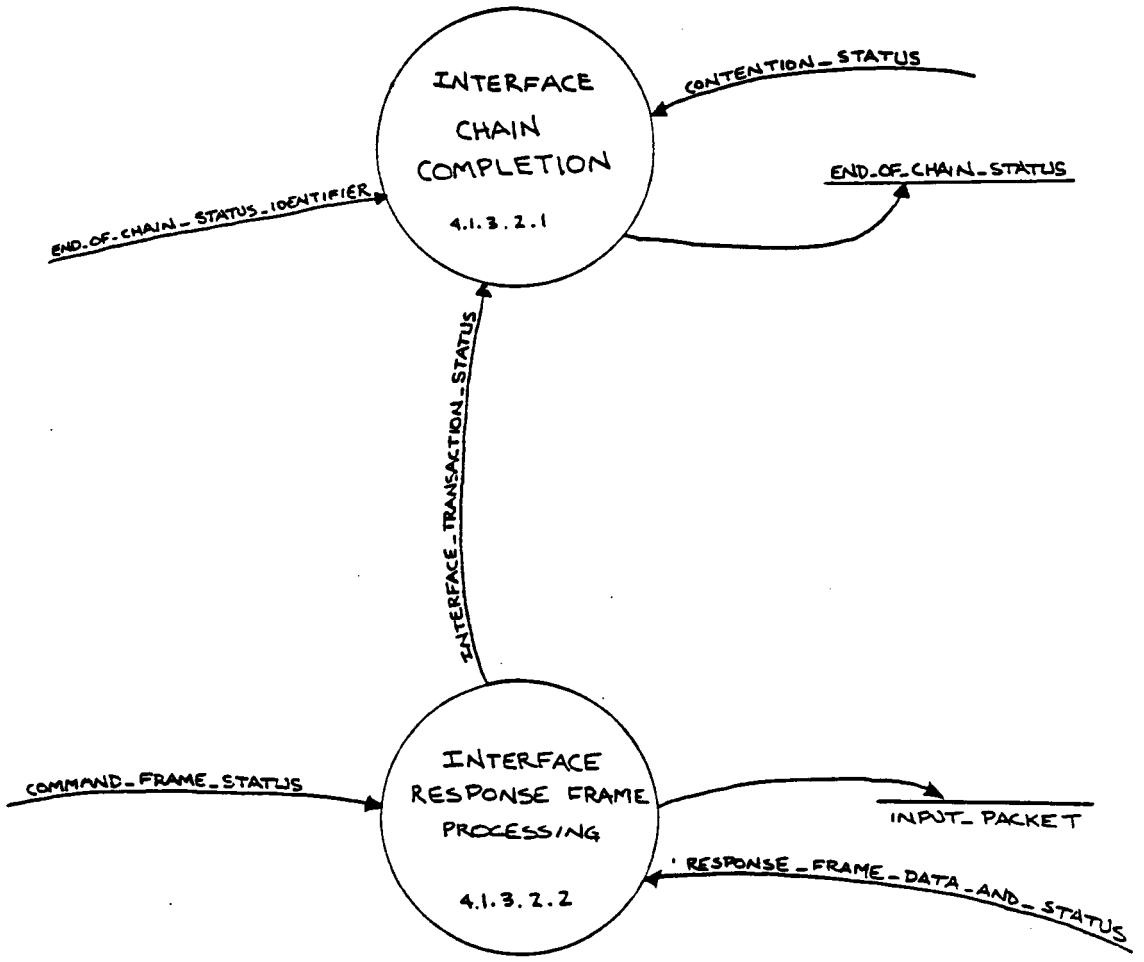


Figure 2-12. Interface Initial Processing - 4.1.3.1

This page is intentionally blank.



PRECEDING PAGE BLANK NOT FILMED

Figure 2-13. Interface Completion Processing - 4.1.3.2

This page is intentionally blank.

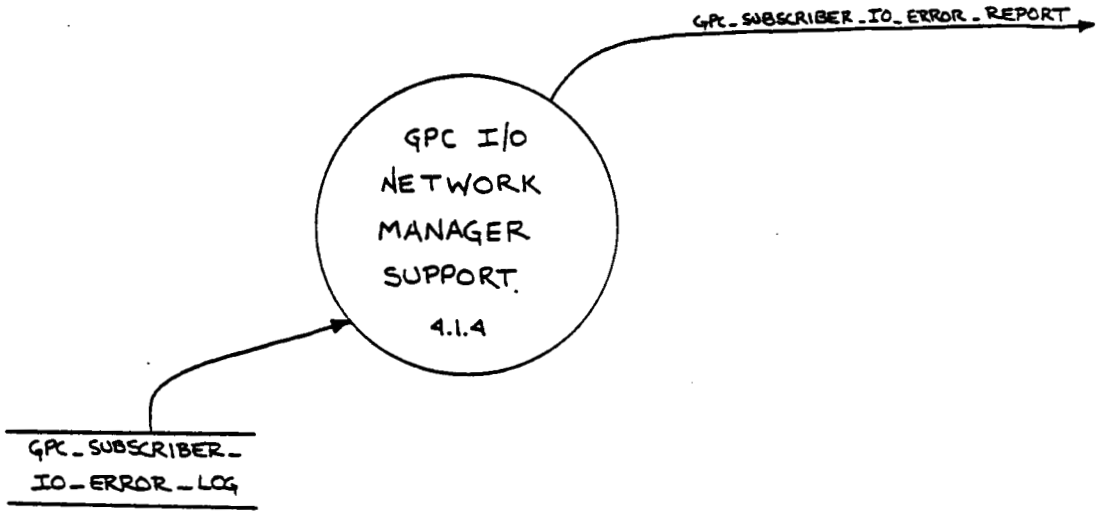


Figure 2-14. GPC I/O Network Manager Support - 4.1.4

This page is intentionally blank.

2.2 I/O Network Manager

I/O Network Manager processing is divided into three groups: Network Control, Network Monitoring, and Network Status Reporting. The data flow diagrams are in Figure 2-16 on page 2-35 through Figure 2-22 on page 2-47 . The hierarchical relationship of these flow diagrams and processes is illustrated in Figure 2-15 on page 2-32.

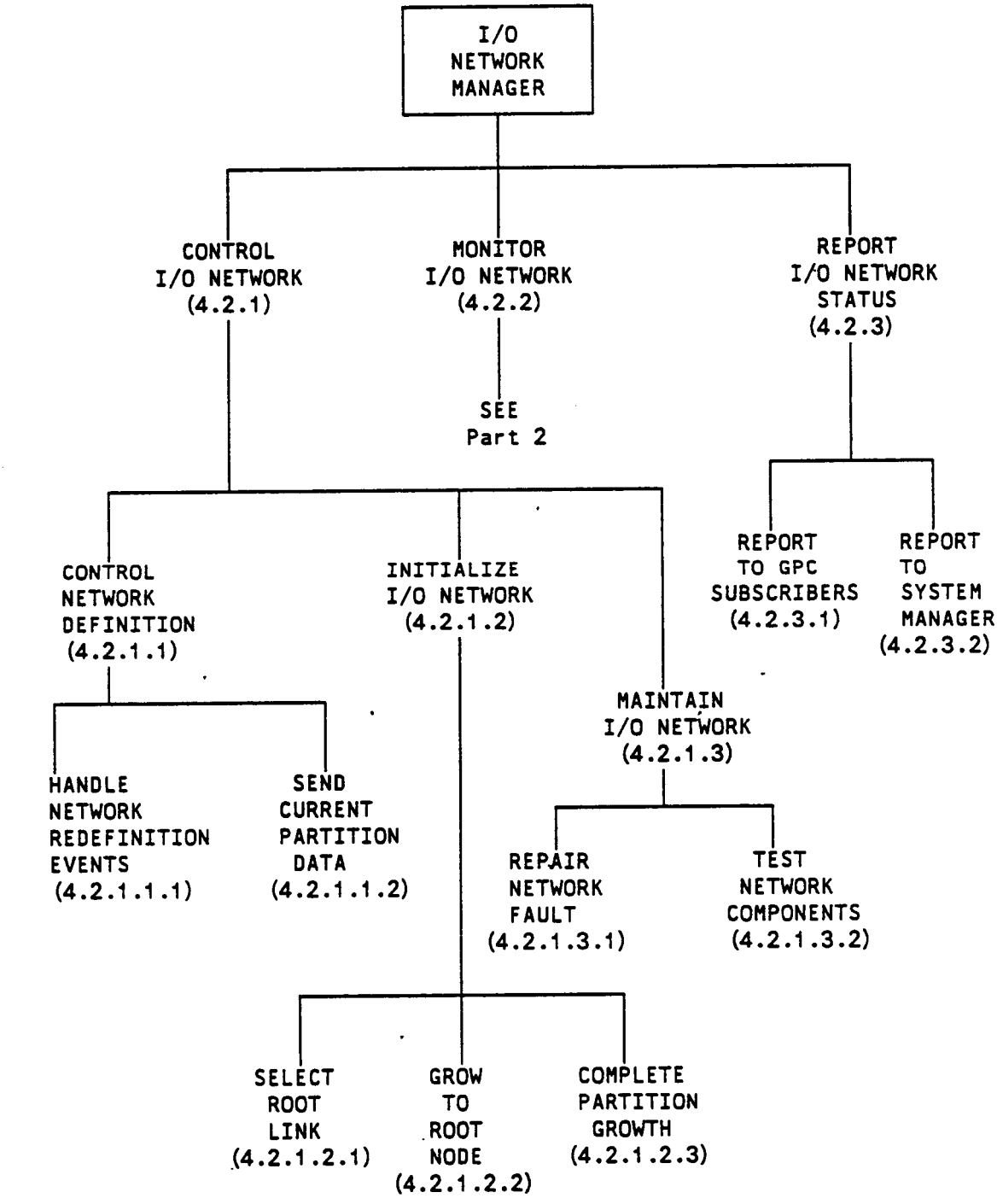


Figure 2-15. I/O Network Manager (Part 1 of 2): Data Flow Diagram Processing Hierarchy

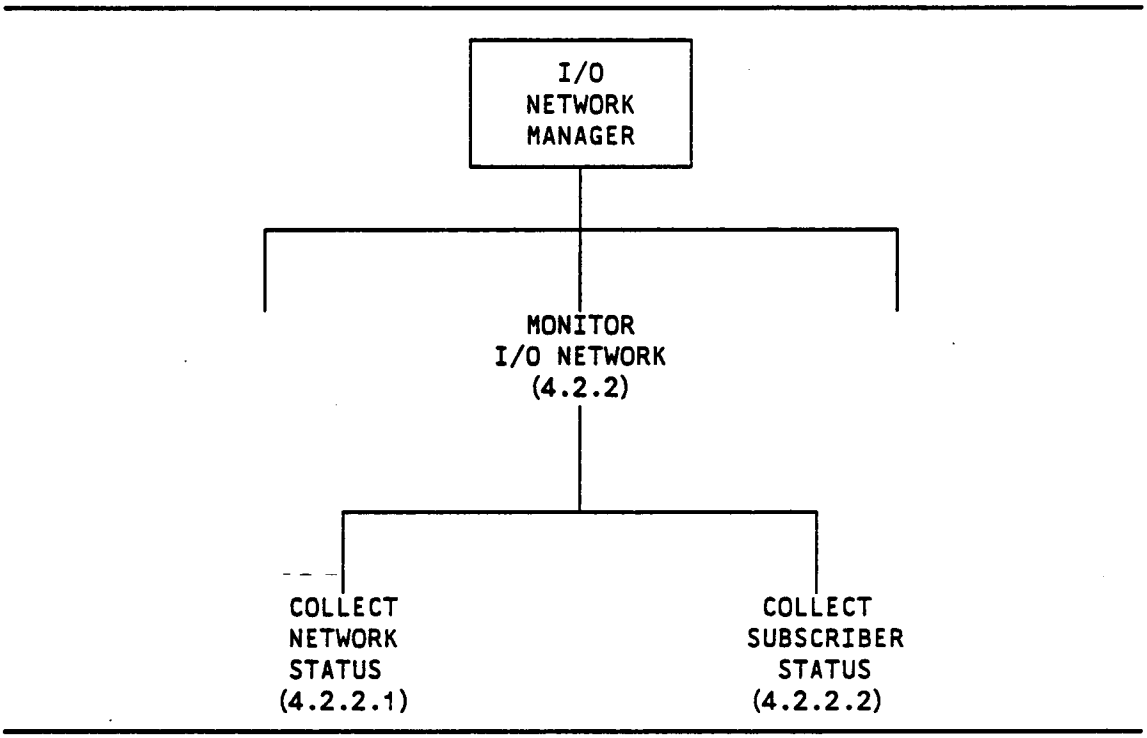


Figure 2-15. I/O Network Manager (Part 2 of 2): Data Flow Diagram Processing Hierarchy

This page is intentionally blank.

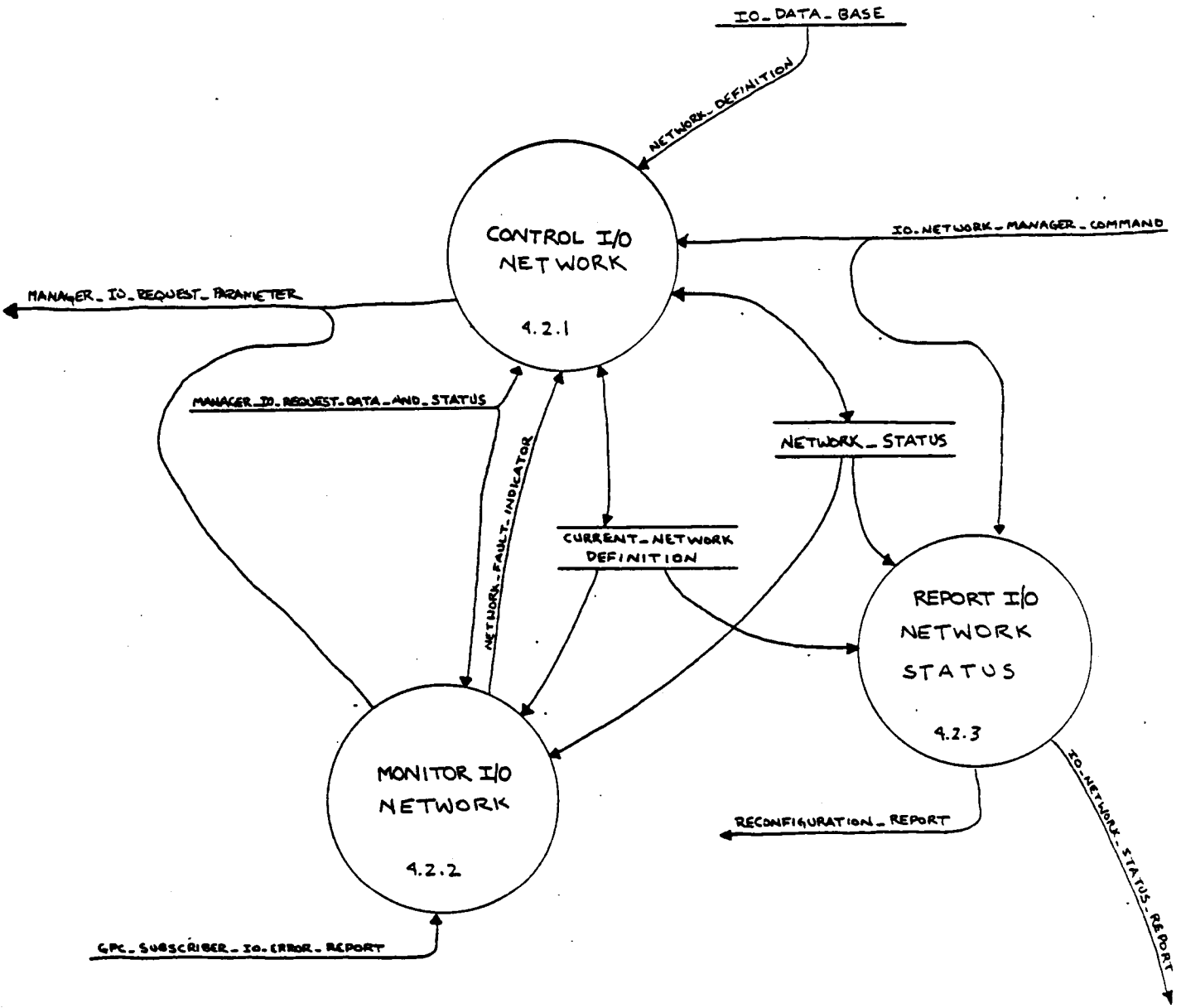


Figure 2-16. I/O Network Manager - 4.2.

This page is intentionally blank.

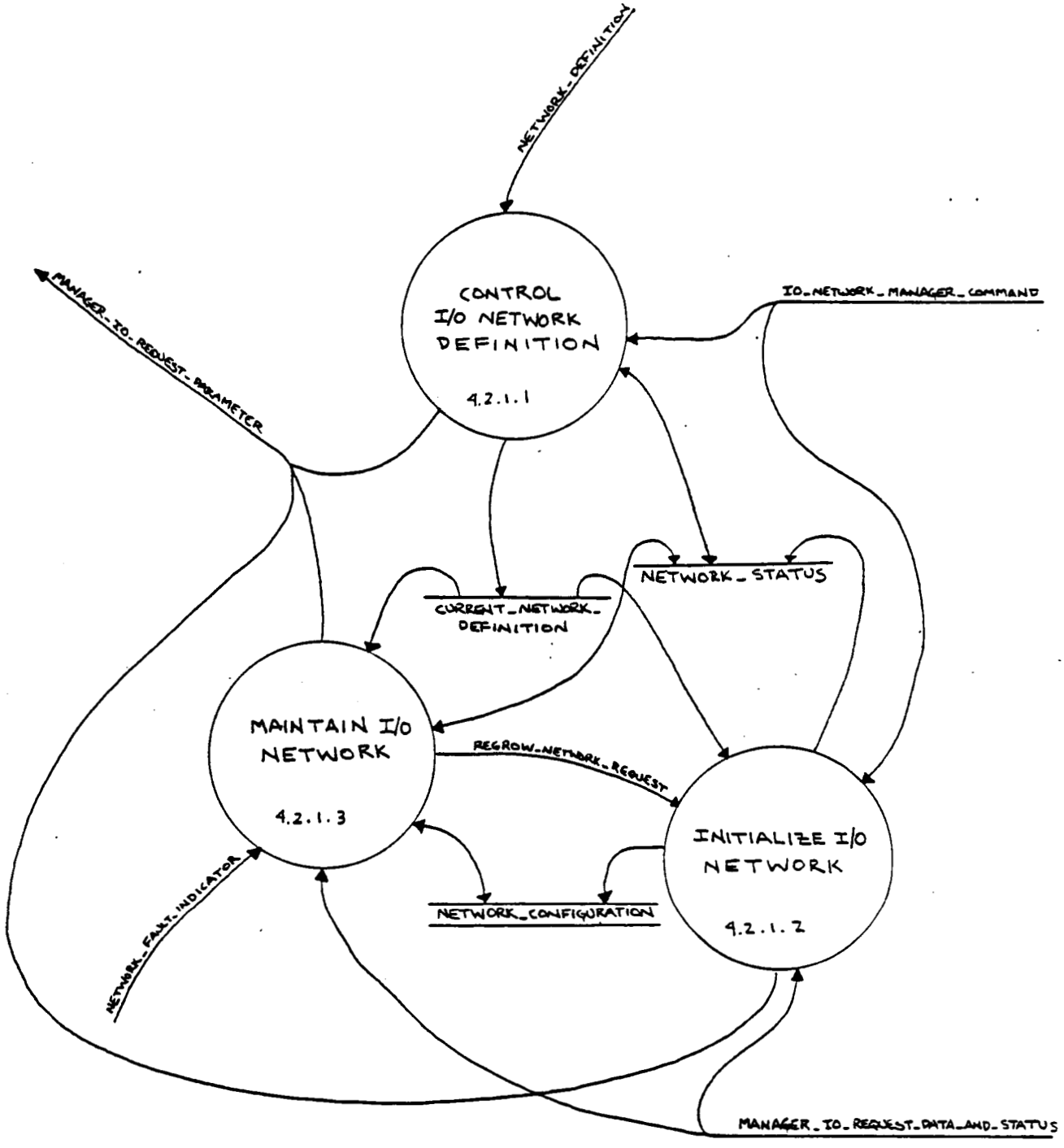


Figure 2-17. Control I/O Network - 4.2.1

This page is intentionally blank.

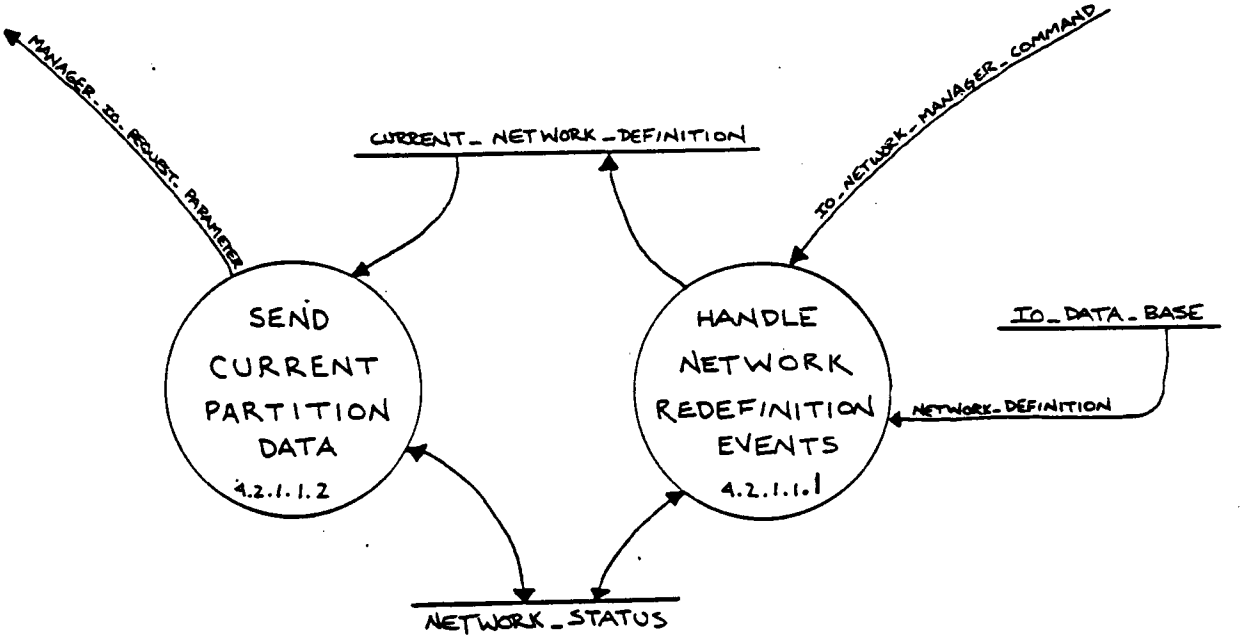


Figure 2-18. Control Network Definition - 4.2.1.1

This page is intentionally blank.

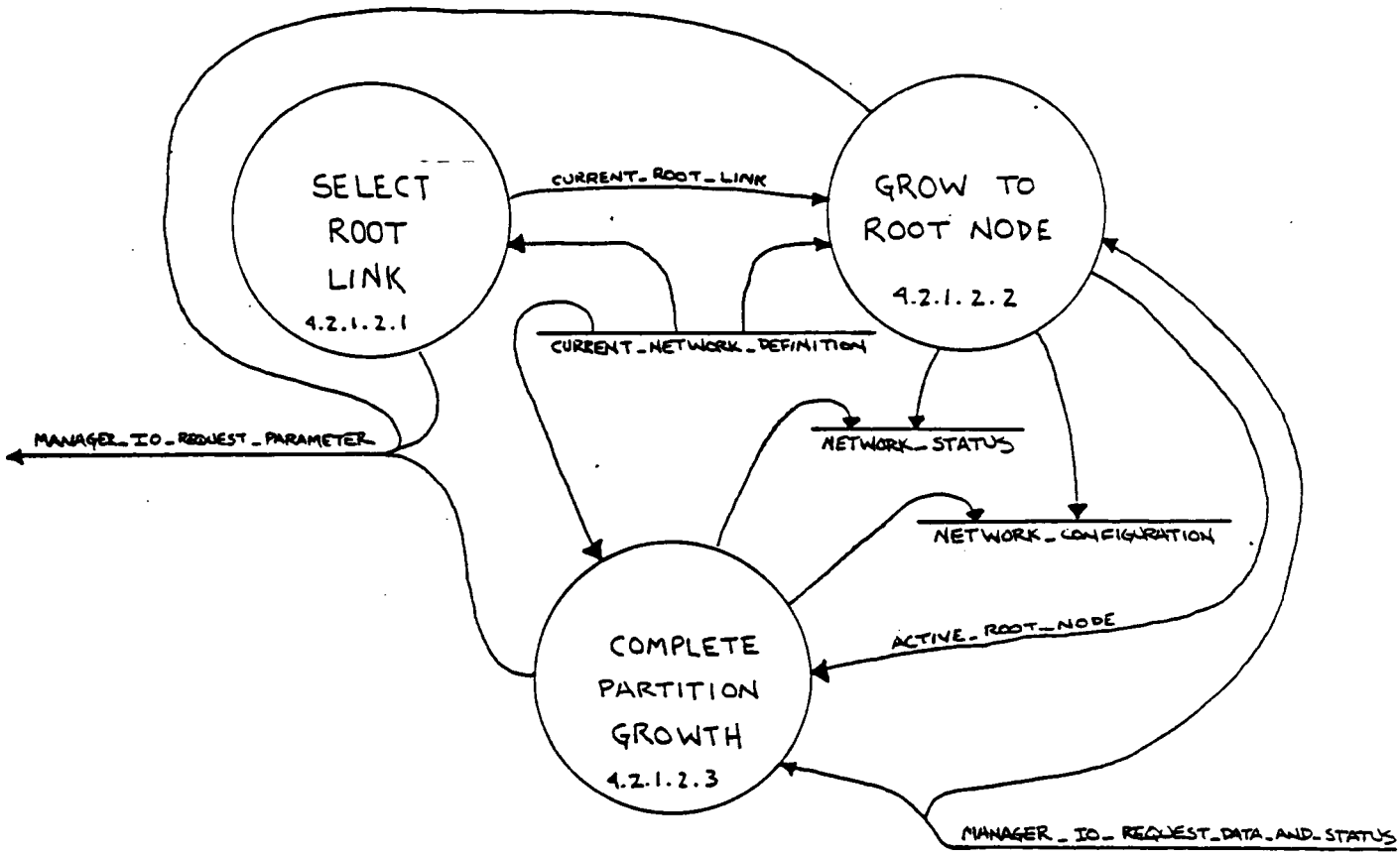
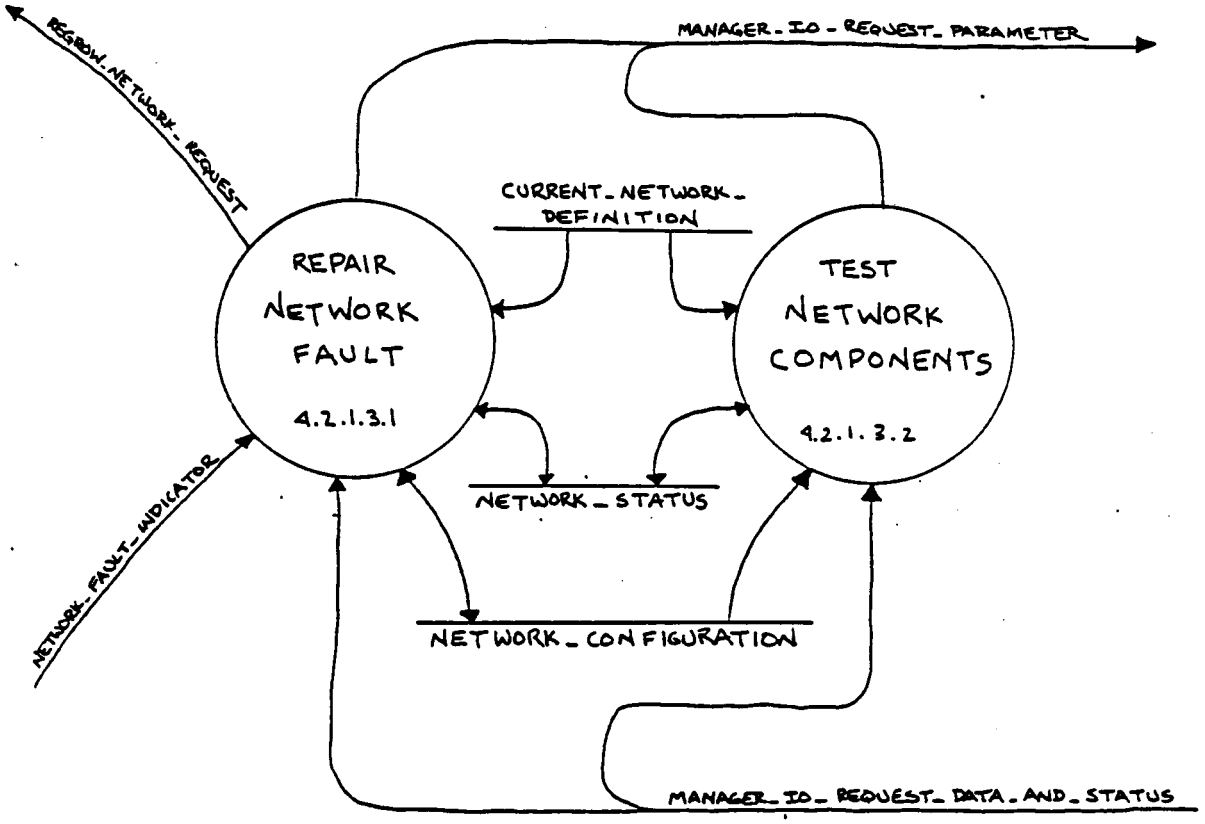


Figure 2-19. Initialize I/O Network - 4.2.1.2

This page is intentionally blank.



PRECEDING PAGE BLANK NOT FILMED

Figure 2-20. Maintain I/O Network - 4.2.1.3

This page is intentionally blank.

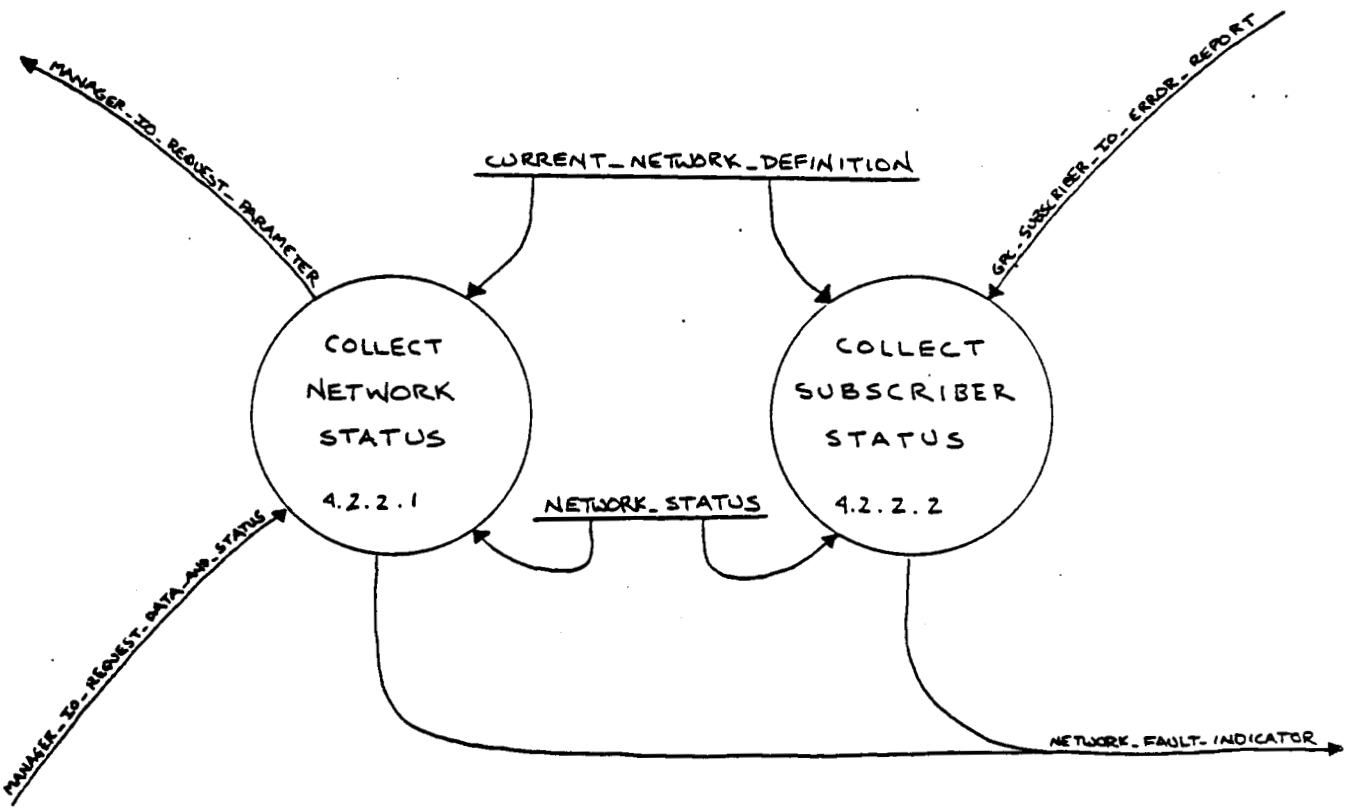


Figure 2-21. Monitor I/O Network - 4.2.2

This page is intentionally blank.

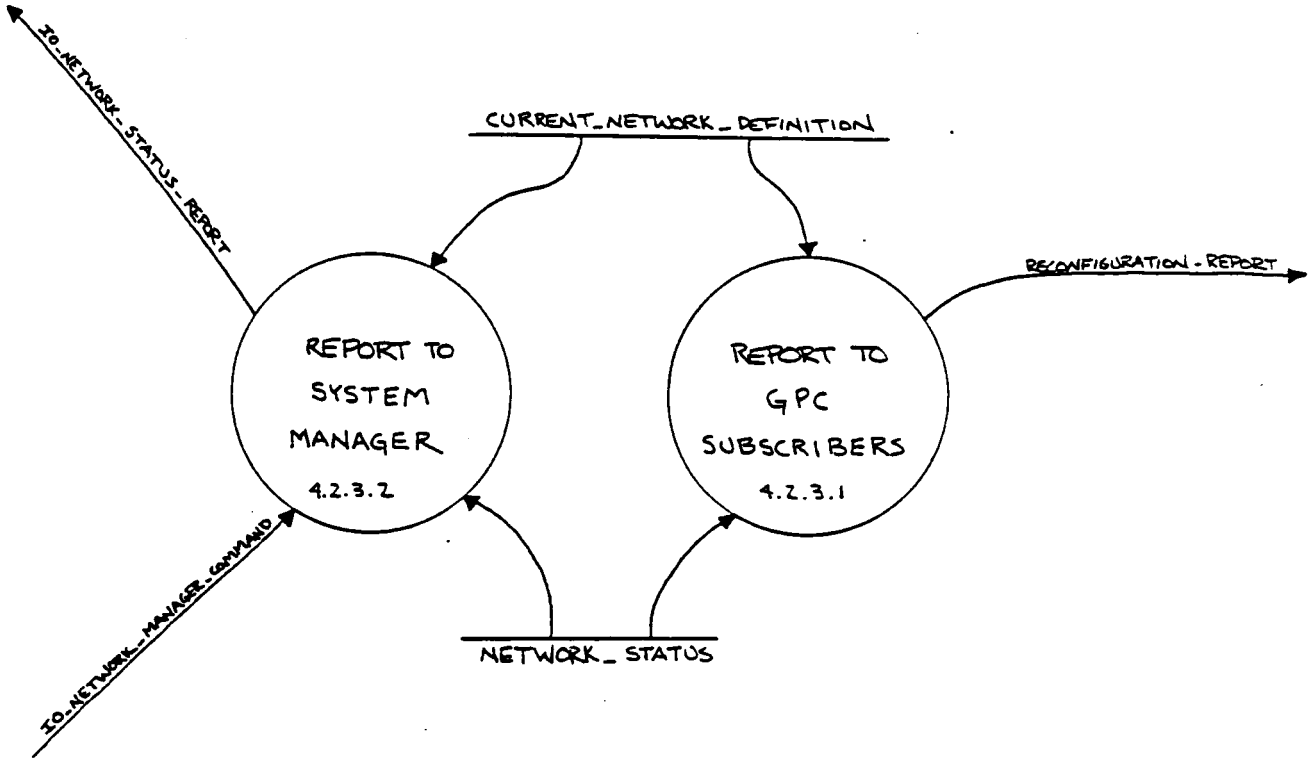


Figure 2-22. Report I/O Network Status - 4.2.3

This page is intentionally blank.

SECTION 3

PROCESS DESCRIPTIONS

This section contains a description of each process identified in the data flow diagrams. The descriptions are in a standard format which is described in appendix " B. Process Description Format Explanation."

3.1 I/O User Communication Services Processes

Process Name: I/O User Communication Services
Reference Number: 4.1
Identifier: IO_System_Services.-
IO_User_Communication_Services
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, Chapter 3

Inputs:

- GPC_Subscriber_Command .from System Manager (1.)
- IO_Request_Parameter from Application Functions
- Response_Frame from DIUs
- IO_Data_Base
- Reconfiguration_Report from I/O Network Manager (4.2)
- Manager_IO_Request_Parameter from I/O Network Manager (4.2)

Outputs:

- Local_IO_Status to Local System Services (3.)
- IO_Request_Data_And_Status to Application Functions
- Wait_Request to Local Operating System
- Command_Frame to DIUs
- GPC_Subscriber_IO_Error_Report to I/O Network Manager (4.2)
- Manager_IO_Request_Data_And_Status to I/O Network Manager (4.2)
- GPC_Subscriber_IO_Error_Log to System Manager (1.)

Notes: This process must exist at each processing site which provides I/O services to resident functions.

Description:

This process provides communication to DIUs and Nodes throughout the system. It handles all requests for the following communication services:

- Communication with nodes and DIUs including automatic bypassing of transactions which repeatedly cause errors,
- Transaction selection for specified transactions to be performed within an I/O request,
- Clearing of the Bypass for all transactions on the network,
- Transaction selection for all transactions used by a particular function,
- Updating a network partitioning definition, and
- Switching the root link(s) (I/O Interface (4.1.3)) connecting an I/O network to a processing site.

The above services are implemented by the following subprocesses:

(1) I/O Request Processing

(2) Chain Processing

(3) I/O Interface

(4) GPC I/O Network Manager Support

I/O Request Processing (4.1.1) handles all requests and coordinates services affecting one or more networks.

Each instance of Chain Processing (4.1.2) implements services as they apply to a particular I/O network at a processing site. It also coordinates the I/O Interfaces (4.1.3) for that I/O network.

One or more I/O Interfaces (4.1.3) connect a processing site to an I/O network. This process implements the actual communication between the processing site and the DIUs and/or nodes connected to the I/O network as specified by Chain Processing (4.1.2).

An instance of GPC I/O Network Manager Support (4.1.4) independently coordinates the reporting of GPC_Subscriber_IO_Error_Log information to a particular I/O Network Manager (4.2).

Process Name: I/O Request Processing
Reference Number: 4.1.1
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
IO_Request_Processing
Build: 3

Requirements Reference: 3

Inputs:

- IO_Data_Base.IO_Request_Definition
- GPC_Subscriber_Command from System Manager (1.)
- IO_Request_Parameter from Application Function
- Manager_IO_Request_Parameter from I/O Network Manager (4.2)
- Reconfiguration_Report from I/O Network Manager (4.2)
- Chain_Queue from Chain Processing (4.1.2)
- Chain_Data_And_Status from Chain Processing (4.1.2)

Outputs:

- Wait_Request to Local Operating System
- IO_Request_Data_And_Status to Application Functions
- Manager_IO_Request_Data_And_Status to I/O Network Manager (4.2)
- Chain_Queue to Chain Processing (4.1.2)

Notes: There is one instance of this process for each instance of I/O User Communication Services (4.1).

Description:

This process coordinates service requests from I/O Network Managers (4.2) and service requests from Application Functions that pertain to one or more I/O networks. Each service request is transformed into one or more elements on one or more Chain_Queue(s) to be processed by instances of Chain Processing (4.1.2). Status and data resulting from this processing are collected via Chain_Queue and Chain_Data_And_Status.

The above processing is accomplished via two subprocesses:

(1) Request Initiation Processing

(2) Request Completion Processing

Request Initiation Processing (4.1.1.1) transforms service requests that are input as IO_Request_Parameter, Manager_IO_Request_Parameter, GPC_Subscriber_Command, or Reconfiguration_Report into elements on the appropriate Chain_Queue(s).

Request Completion Processing (4.1.1.2) collects data and status from Chain_Queue and Chain_Data_And_Status inputs and transforms them into IO_Request_Data_And_Status and Manager_IO_Request_Data_And_Status.

Process Name: Request Initiation Processing
Reference Number: 4.1.1.1
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
IO_Request_Processing.-
Request_Initiation_Processing
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 2, Paragraph 2 and 3 and sections 3.1.4.1, 3.1.4.2, 3.1.4.4.1, 3.3

Inputs:

- IO_Data_Base.IO_Request_Definition
- IO_Request_Parameter from Application Functions
- GPC_Subscriber_Command from System Manager (1.)
- Manager_IO_Request_Parameter from I/O Network Manager (4.2)
- Reconfiguration_Report from I/O Network Manager (4.2)
- Chain_Queue from Request Completion Processing (4.1.1.2) and Chain Processing (4.1.2)

Outputs:

- IO_Request_Data_And_Status to Application Functions
- Wait_Request to Local Operating System
- Chain_Queue to Chain Processing (4.1.2) and Request Completion Processing (4.1.1.2)
- Manager_IO_Request_Data_And_Status to I/O Network Manager (4.2)

Notes: Processing by this process should not lock out processing by other processes (such as Sequencer (4.1.2.1.1) and Request Completion Processing (4.1.1.2)) due to accesses to Chain_Queue data. This may be implemented by semaphores, a monitoring process, or some other mechanism to control access to Chain_Queue data.

Description:

This process transforms each request for service into elements on one or more Chain_Queue(s). Services are then implemented as the elements from the Chain_Queue(s) are processed by their corresponding Chain Processing (4.1.2) processes. Each Chain_Queue corresponds to an instance of Chain Processing (4.1.2). Each instance of Chain Processing corresponds to a specific I/O network.

Communication with nodes and/or DIUs is requested via IO_Request_Parameter or Manager_IO_Request_Parameter. Either input specifies the same information. A Service_Identifier indicates the

request is an IO_Service_Request. An IO_Request_Identifier indicates a specific IO_Request_Definition within the IO_Data_Base. This definition directs how IO_Request_Data from IO_Request_Parameter or Manager_IO_Request_Parameter should be distributed between one or more Transaction_Queue_Elements, hence, how the data should be distributed between the transactions performed on each I/O network. The definition also indicates the Chain_Queue in which each element should be inserted (one Chain_Queue per element). Each element is inserted into its Chain_Queue according to the priority implied by the input (Manager_IO_Request_Parameter having a higher priority than IO_Request_Parameter) or according to the IO_Request_Priority explicitly specified by IO_Request_Parameter. The definition also specifies whether a Wait_Request should be made for the process which made the request or the chain completion indicators for the request should be initialized to "Not_Finished_Yet". (If the process is caused to wait, it is released by Request Completion Processing (4.1.1.2) when the request is completed.)

Transaction selection is also requested via IO_Request_Parameter or Manager_IO_Request_Parameter. A Service_Identifier indicates the request is a Transaction_Selection_Request. An IO_Request_Identifier indicates a specific IO_Request_Definition within the IO_Data_Base. Selection_Queue_Elements are constructed from the Chain_Identifiers, Transaction_Identifiers, and Selection components of the input and inserted into Chain_Queues as directed by the IO_Request_Definition. These elements are assumed to have a priority higher than the priority of their corresponding IO_Service_Request, i.e., Chain_Queue elements created for a Transaction_Selection_Request will always precede elements created for a IO_Service_Request that has the same Request_Identifier.

Clearing Bypass for all transactions on a network is requested via the Reconfiguration_Report input. This will be implemented by marking a chain of transactions for clearing, the Bypass for each transaction to be cleared the next time the chain is executed. The Network_Identifier specified by this input indicates the Chain_Queue on which to place the Bypass_Clear_Queue_Element specified by the input. This element specifies that all transactions on this network should have their error counts and bypasses initialized to zero and "No", respectively.

Transaction selection for all transactions used by an Application Function may be requested either via IO_Request_Parameter or via GPC_Subscriber_Command. IO_Request_Parameter specifies a Service_Identifier value of "Application_Initialization_Request". Either input specifies a Function_Identifier specifying which function is to be initialized. Function_Identifier indicates a group of IO_Request_Definitions in IO_Data_Base. These definitions provide the Selection_Defaults used to construct Selection_Queue_Elements for each I/O network accessed by the Application Function and to place these elements on the correct Chain_Queues.

Updates to network partition definitions are requested via Manager_IO_Request_Parameter. The Service_Identifier component spec-

ifies Partition_Update_Request. The Network_Identifier component specifies which Chain_Queue should receive the Network_Partition_Queue_Element which is constructed from the Root_Link_Identifiers, Node_Identifiers, and DIU_Identifiers listed in the remainder of the input that specify which Nodes and DIUs may be accessed via which root links.

Root link switching is requested via Manager_IO_Request_Parameter. The Service_Identifier component specifies Root_Link_Control_Request. The Network_Identifier component specifies which Chain_Queue should receive the Root_Link_Queue_Element, i.e., for which network the root link should be switched. The queue element specifies whether or not automatic root link switching should be inhibited and which root links should be used according to the Inhibit and Root_Link_Identifier components of the input.

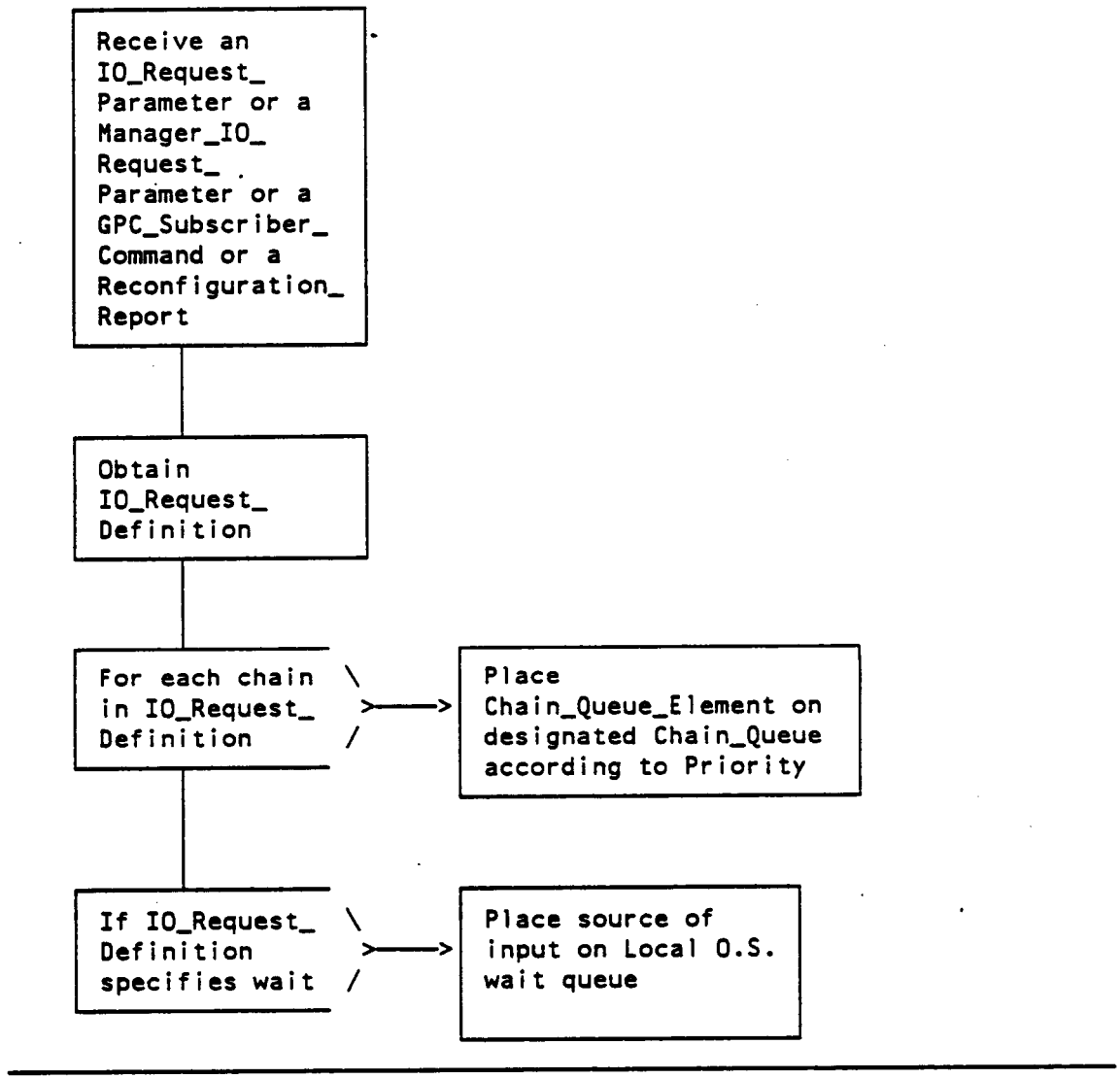


Figure 3-1. Request Initiation Processing

Process Name: Request Completion Processing
Reference Number: 4.1.1.2
Identifier: IO_System_Services.-
IO_User_Communication_Services-
IO_Request_Processing.-
Request_Completion_Processing
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, sections 3.1.4.1, 3.1.4.2, 3.1.4.4.1

Inputs:

- IO_Data_Base.IO_Request_Definition
- Chain_Data_And_Status from Chain Processing (4.1.2)
- Chain_Queue from Request Initiation Processing (4.1.1.1) and Chain Processing (4.1.2)

Outputs:

- Wait_Request to Local Operating System
- IO_Request_Data_And_Status to Application Functions
- Manager_IO_Request_Data_And_Status to I/O Network Manager (4.2)
- Chain_Queue to Request Initiation Processing (4.1.1.1) and Chain Processing (4.1.2)

Notes: Processing by this process should not lock out processing by other processes (such as Request Initiation Processing (4.1.1.1) and Sequencer (4.1.2.1.1)) due to accesses and references to Chain_Queue data. This may be implemented by semaphores, a monitoring process, or some other mechanism to control access to Chain_Queue data.

Description:

This process completes processing of requests for communication to Nodes (4.1.5) and DIUs by collecting data and status and releasing processes that were placed on the local wait queue at the request of Request Initiation Processing (4.1.1.1). It is initiated by a signal from Chain Processing (4.1.2).

When Chain_Data_And_Status is received, the element at the top of the corresponding Chain_Queue is examined. The Chain_Identifier is used to identify the IO_Request_Definition that specified the element for Request Initiation Processing (4.1.1.1).

If the IO_Request_Definition indicates that the original request was a manager I/O request for service:

- Manager_IO_Request_Data_And_Status is updated to reflect the values of Chain_Data_And_Status,

- Manager_IO_Request_Data_And_Status is updated to reflect the values of Root_Link_Status from the Chain_Queue element,

- The element is removed from the Chain_Queue,

- If the Network Manager (4.2) making the request was placed on the local wait queue at the request of Request Initiation Processing (4.1.1.1), this process issues a Wait_Request to the Local Operating System to remove the manager from the local wait queue. Otherwise, the manager can determine that the I/O request has completed by examining the status value for the chain in Manager_IO_Request_Data_And_Status.

If the IO_Request_Definition indicates that the original request was from an Application-Function:

- The element is removed from the Chain_Queue.

- IO_Request_Data_And_Status is updated with the values of Chain_Data_And_Status, as specified by the IO_Request_Definition.

- If function was placed on the local wait queue at the request of Request Initiation Processing (4.1.1.1), the process determines whether or not all chains for the I/O request have been processed. If they have, the process requests that the Application Function be released from the local wait queue by issuing a Wait_Request to Local Operating System. Otherwise, the requesting function can determine that the chain has completed by examining the status value for the chain in IO_Request_Data_And_Status.

Process Name: Chain Processing
Reference Number: 4.1.2
Identifier: IO_System_Services.-
IO_User_Communication_Services-
Chain_Processing
Build: 3

Requirements Reference: (See subprocesses)

Inputs:

- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- Chain_Queue from I/O Request Processing (4.1.1)
- End_Of_Chain_Status from I/O Interface (4.1.3)
- Input_Packet from I/O Interface (4.1.3)
- Transaction_Configuration_Data_Base from I/O Interface (4.1.3)
- Current_Network_Partition_Definition from I/O Interface (4.1.3)

Outputs:

- Local_IO_Status to Local System Services (3.)
- Chain_Data_And_Status to I/O Request Processing (4.1.1)
- Chain_Queue to I/O Request Processing (4.1.1)
- Chain_Initiation_Command to I/O Interface (4.1.3)
- Current_Network_Partition_Definition to I/O Interface (4.1.3)
- Output_Packet to I/O Interface (4.1.3)
- Transaction_Configuration_Data_Base to I/O Interface (4.1.3)
- GPC_Subscriber_IO_Error_Log to GPC I/O Network Manager Support (4.1.4)

Notes:

An instance of this process must exist at each processing site for each I/O network that is accessible at the processing site.

Queue Processing (4.1.2.1) must not interrupt Chain Interface (4.1.2.3).

End Of Chain Monitor (4.1.2.2.2.1) may interrupt Queue Processing (4.1.2.1).

Description:

This process sequentially processes the elements from the Chain_Queue corresponding to its I/O network. Each element initiates one of the following services on the network:

- Communication with nodes or DIUs including automatic bypassing of transactions which repeatedly cause errors,

- Selection of which transactions to perform within a chain of transactions on the network,
- Clearing of Bypass for all transactions in selected chains on the network,
- Updating the network partitioning definition, and
- Switching the root link(s) (I/O Interface (4.1.3)) connecting the I/O network to a processing site.

The above services are implemented by three subprocesses:

(1) Queue Processing

(2) Chain Completion Processing

(3) Chain Interface

Queue Processing (4.1.2.1) initiates all requests for service on a particular I/O network. It implements selection of transactions, clearing of Bypass for transactions, and updates to the Current_Network_Partition_Definition. It initiates Chain Completion Processing (4.1.2.2) and Chain Interface (4.1.2.3) to implement communication with nodes and DIUs, signaling I/O Request Processing (4.1.1) when the communication processing has been completed. It coordinates processing between Chain Completion Processing (4.1.2.2) and Chain Interface (4.1.2.3) to implement the switching of root links.

Chain Completion Processing (4.1.2.2) monitors the I/O Interfaces (4.1.3) to complete the processing of a chain of transactions, records errors that occurred during communications, and decides when errors indicate that a root link should be switched.

Chain Interface (4.1.2.3) interfaces Queue Processing (4.1.2.1) with I/O Interface (4.1.3) to initiate chains of transactions on the network and to switch root links.

Process Name: Queue Processing
Reference Number: 4.1.2.1
Identifier: IO_System_Services.-
IO_User_Communication_Services-
Chain_Processing.Queue_Processing
Build: 3

Requirements Reference: (See subprocesses)

Inputs:

- Chain_Queue
- Current_Network_Partition_Definition
- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- IO_Data_Base.Limits_Definition
- Chain_Completion_Status from Chain Completion Processing (4.1.2.2)
- Transaction_Configuration_Data_Base

Outputs:

- Chain_Queue
- Local_IO_Status to Local System Services (3.)
- Current_Network_Partition_Definition
- Output_Packet
- Transaction_Configuration_Data_Base to I/O Interface (4.1.3) and Chain Completion Processing (4.1.2.2)
- Chain_Timeout_Value to Chain Completion Processing (4.1.2.2)
- Activate_Chain to Chain Interface (4.1.2.3)
- Root_Link_Command to Chain Interface (4.1.2.3)

Notes: The Activate_Chain and Root_Link_Command data flows must be nonobtrusive (asynchronous) to Chain Interface (4.1.2.3). In other words, the implementation should not interrupt this process.

Description:

This process implements the following services for a single I/O network:

- Selection of which transactions to perform within a chain of transactions on the network,
- Clearing of Bypass for all transactions in selected chains on the network, and
- Updating the network partition definition.

It also initiates the following services:

- Communication with nodes or DIUs, coordinating with Chain Interface (4.1.2.3) and Chain Completion Processing (4.1.2.2) to

implement automatic retries and automatic switching of root links via this service.

- Switching the root link(s) (I/O Interface (4.1.3)) connecting the I/O network to a processing site.

The above is accomplished via six subprocesses:

(1) Sequencer

(2) Chain Initiation

(3) Root Link

(4) Bypass Clear

(5) Transaction Selection

(6) Network Partition

Sequencer (4.1.2.1.1) invokes the other subprocesses according to the top element found in Chain_Queue. It also invokes Root Link (4.1.2.1.3) when requested by Chain Completion Processing (4.1.2.2) for automatic root link switching and invokes Chain Initiation (4.1.2.1.2) for automatic retry of communications to nodes or DIUs.

Chain Initiation (4.1.2.1.2) initiates communications to nodes or DIUs and clears the Bypass for all transactions in chains that are marked for bypass clearing in the Transaction_Configuration_Data_Base.

Root Link (4.1.2.1.3) chooses root links and initiates root link switching.

Bypass Clear (4.1.2.1.4) marks chains of transactions for clearing of their bypass states upon the next occurrence of the chain.

Transaction Selection (4.1.2.1.5) sets up the selection of transactions to be performed in specified chains.

Network Partition (4.1.2.1.6) updates Current_Network_Partition_Definition.

Process Name: Sequencer
Reference Number: 4.1.2.1.1
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Queue_Processing.-
Sequencer
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 2, paragraph 3 and section 3.1.5

Inputs:

- Chain_Queue
- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- IO_Data_Base.Limits_Definition
- Chain_Completion_Status from Chain Completion Processing (4.1.2.2)
- Root_Link_Status from Root Link (4.1.2.1.3)

Outputs:

- Chain_Queue
- Local_IO_Status
- Transaction_Queue_Element to Chain Initiation (4.1.2.1.2)
- Root_Link_Queue_Element to Root Link (4.1.2.1.3)
- Bypass_Clear_Queue_Element to Bypass Clear (4.1.2.1.4)
- Selection_Queue_Element to Transaction Selection (4.1.2.1.5)
- Network_Partition_Queue_Element to Network Partition (4.1.2.1.6)

Notes: Processing by this process should not lock out processing by other processes (such as Request Initiation Processing (4.1.1.1) and Request Completion Processing (4.1.1.2)) due to accesses and references to Chain_Queue data. This may be implemented by semaphores, a monitoring process, or some other mechanism to control access to Chain_Queue data.

Description:

This process initiates the following services for its I/O network:

- Communication with nodes or DIUs,
- Switching the root link(s) (I/O Interface (4.1.3)) connecting the I/O network to a processing site,
- Clearing the Bypass for all transactions in selected chains on the network,

- Selection of which transactions to perform within a chain of transactions on the network, and
- Updating the network partition definition.

No service is initiated until the previous service has completed. When no service is in progress and the Chain_Queue is not empty, the top element of the queue is selected to determine which service to initiate. The element also indicates the parameters to be used to implement the service.

Communication to Nodes or DIUs is initiated by passing the Chain_Queue element as a Transaction_Queue_Element to Chain Initiation (4.1.2.1.2). The element is not removed from Chain_Queue by this process. (It is removed by Request Completion Processing 4.1.1.2 after the service is completed.) Completion of the chain of transactions is indicated by Chain_Completion_Status which has three possible values (see also End of Chain I/O Error Processing (4.1.2.2.2.2)):

"Next_Chain" indicates that processing is completed. The process sets the Chain_Complete data item in the Chain_Queue element to "Chain_Complete". If the communication request was from a Network Manager (as indicated IO_Chain_Definition when indexed by the Chain_Identifier in the Chain_Queue element), the Root_Link_Identifiers for the currently active root links are added to the Chain_Queue element. (This is why the element is not removed. It must remain on the Chain_Queue for outputting data items to Request Completion Processing 4.1.1.2.) In any case, processing halts until a new element appears at the top of the Chain_Queue (See Request Completion Processing (4.1.1.2)).

"Switch_Root_Link_And_Next_Chain" indicates that, before completing processing, the root link(s) should be switched. If the communication request came from an I/O Network Manager (4.2) (see "Next_Chain" above) and the Manager_IO_Request_Parameter (Root_Link_Control_Request) indicates "Inhibit_Switching", the switch request is treated as if it were "Next_Chain". Otherwise, the root link is switched as described below and processing of the chain is completed by assigning Root_Link_Identifiers and Chain_Compete (see the description of "Next_Chain", above).

There are three types of internal counters for root link switching: one to count how many times a particular root link is switched, one to count how many times all the root links for a partition have been switched as a whole, and one to count how may times a chain has been retried. The retry counter is always initialized to zero when a new communication is requested via a Chain_Queue element. Each time a root link is to be switched, one or more counters are incremented by one.

If the switch will cause a root link counter to exceed Switching_Limit.Single_Link_Log_Limit (found in Limits_Definition), this fact is logged with the current time and Chain_Identifier, the counter is initialized to zero. In this case, the root link will be switched. Processing continues as described in the next paragraph.

If the switch will cause a group counter to exceed Switching_Limit.Rotation_Log_Limit (also found in Limits_Definition), this fact is logged with the current time and Chain_Identifier, the counter is initialized to zero. In this case, the root link will be switched. Processing continues as described in the next paragraph.

If there is another I/O Interface (4.1.3) to use for the given partition, the switch is initiated via Root_Link_Queue_Element which indicates the partition of the I/O network which is to have its root link switched. (The actual switch does not occur if there is no alternative root link; there is no reason to switch from one root link to itself.) After the root link is switched, the new Root_Link_Identifier value indicated by Root_Link_Status is stored.

"Switch_Root_Link_And_Repeat_Chain" indicates that the root link should be switched and the same communication request attempted again. (This status may be caused by the failure of the I/O Interface to win a contention.)

If the communication request came from a Network Manager (4.2) (see "Next_Chain" above) and the Manager_IO_Request_Parameter (Root_Link_Control_Request) indicates "Inhibit_Switching", the root link is not switched, the chain is not repeated, and the Chain_Queue element is assigned values for Root_Link_Identifiers and Chain_Complete (see "Next_Chain", above).

Otherwise, the retry counter is incremented by one. If the retry counter exceeds Switching_Limit.Retry_Limit (located in Limits_Definition), the root link is not switched, the chain is not repeated, and the Chain_Queue element is assigned values for Root_Link_Identifiers and Chain_Complete (see "Next_Chain", above).

Otherwise, processing to switch the I/O Interface (4.1.3) continues as described in "Switch_Root_Link_And_Next_Chain" above.

After the retry counter is incremented and the switch has been performed (or skipped due to the lack of an alternative root link), the chain is repeated reissuing the Transaction_Queue_Element. This causes a new value to be returned for Chain_Completion_Status.

Explicit root link switching is initiated by passing on a Root_Link_Queue_Element. Additional processing includes storing the Root_Link_Identifier indicated by Root_Link_Status and removing the element from the Chain_Queue.

Marking of chains of transactions for bypass clearing is initiated by passing on a Bypass_Clear_Queue_Element. Additional processing includes removing the element from the Chain_Queue.

Selection of transactions to be performed within a chain is initiated by passing on a Selection_Queue_Element. Additional processing includes removing the element from the Chain_Queue.

Updating of the current network partition definition is initiated by passing on a Network_Partition_Queue_Element. Additional processing includes removing the element from the Chain_Queue.

Process Name: Chain Initiation
Reference Number: 4.1.2.1.2
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Queue_Processing.-
Chain_Initiation
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 3.1.4.2 paragraph 1, section 3.3.3

Inputs:

- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- Transaction_Configuration_Data_Base
- Transaction_Queue_Element from Sequencer (4.1.2.1.1)

Outputs:

- Transaction_Configuration_Data_Base
- Output_Packet
- Chain_Timeout_Value to Chain Completion Processing (4.1.2.2)
- Activate_Chain to Chain Interface (4.1.2.3)

Notes: The Activate_Chain data flow must be nonobtrusive (asynchronous) to Chain Interface (4.1.2.3). In other words, the implementation should not interrupt this process.

Description:

This process sets up the chain of transactions to be communicated on the I/O network. It initializes an Output_Packet for each transaction in the chain with data from Transaction_Queue_Element. The number of transactions and packets and their identities are provided by IO_Chain_Definition.

If the Transaction_Configuration_Data_Base indicates that the chain is marked for bypass clearing (Bypass_Clear equals "Clear"), this process clears the Bypass (sets Bypass to "No") and zeros the Transaction_Error_Counter for each transaction in the chain. Both items are part of the Transaction_Configuration_Data_Base.

Finally, the process outputs the Chain_Timeout_Value (from IO_Chain_Definition) to begin Chain Completion Processing (4.1.2.2). Chain_Timeout_Value includes Chain_Identifier. This process also outputs Chain_Identifier via Activate_Chain to begin the Chain Interface (4.1.2.3) process.

Process Name: Root Link
Reference Number: 4.1.2.1.3
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Queue_Processing.-
Root_Link
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.1

Inputs:

- Root_Link_Queue_Element from Sequencer (4.1.2.1.1)
- Current_Network_Partition_Definition from I/O Network Manager (4.2)

Outputs:

- Root_Link_Command to Chain Interface (4.1.2.3)
- Root_Link_Status to Sequencer (4.1.2.1.1)

Notes: The Root_Link_Command data flow must be nonobtrusive (asynchronous) to Chain Interface (4.1.2.3). In other words, the implementation should not interrupt this process.

Description:

This process computes how to switch root links, i.e., which I/O Interface (4.1.3) to enable and which I/O Interface to disable, and implements the switch via Chain Interface (4.1.2.3). The computation is based on the the Root_Link_Identifier(s) specified by Root_Link_Queue_Element or a rotating choice of one of the alternate root links specified by the Current_Network_Partition_Definition. The Root_Link_Identifier(s) selected is communicated to Sequencer (4.1.2.1.1) via Root_Link_Status.

Process Name: Bypass Clear
Reference Number: 4.1.2.1.4
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Queue_Processing.-
Bypass_Clear
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 3.3.3

Inputs:

- Bypass_Clear_Queue_Element from Sequencer (4.1.2.1.1)

Outputs:

- Transaction_Configuration_Data_Base

Notes: None

Description:

This process marks the Bypass_Clear flag in Transaction_Configuration_Data_Base for each chain indicated by Chain_Identifier in Bypass_Clear_Queue_Element.

Process Name: Transaction Selection
Reference Number: 4.1.2.1.5
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Queue_Processing.-
Transaction_Selection
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 3.3.2

Inputs:

- Selection_Queue_Element from Sequencer (4.1.2.1.1)

Outputs:

- Transaction_Configuration_Data_Base

Notes: None

Description:

This process latches the Selection indicators in Transaction_Configuration_Data_Base to "Select" or "Skip" as specified for each transaction specified in Transaction_Queue_Element.

The operation of latching a transaction Selection indicator to "Select" also causes the Bypass to be cleared and the Transaction_Error_Counter to be zeroed. Both of these items also are in Transaction_Configuration_Data_Base.

Process Name: Network Partition
Reference Number: 4.1.2.1.6
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Queue_Processing.-
Network_Partition
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.2

Inputs:

- Network_Partition_Queue_Element from Sequencer (4.1.2.1.1)

Outputs:

- Current_Network_Partition_Definition to Root Link (4.1.2.1.3) and I/O Interface (4.1.3) and Chain Completion Processing (4.1.2.2)

Notes: None

Description:

This process updates the Current_Network_Partition_Definition. Network_Partition_Queue_Element provides the list new assignments of nodes and DIUs to I/O Interfaces (4.1.3).

Process Name: Chain Completion Processing
Reference Number: 4.1.2.2
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Chain_Completion_Processing
Build: 3

Requirements Reference: (See subprocess descriptions below.)

Inputs:

- Current_Network_Partition_Definition
- End_Of_Chain_Status
- Input_Packet
- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- IO_Data_Base.Limits_Definition
- Transaction_Configuration_Data_Base
- Chain_Timeout_Value from Queue Processing (4.1.2.1)

Outputs:

- GPC_Subscriber_IO_Error_Log
- Transaction_Configuration_Data_Base
- Chain_Data_And_Status to I/O Request Processing (4.1.1)
- Chain_Completion_Status to Queue Processing (4.1.2.1)

Notes: This process may be interrupted by either the Chain Interface (4.1.2.3) process or an internal timer interrupt.

Description:

This process logs errors that occur during a communication service, produces the data and status resulting from the service, and indicates to Queue Processing (4.1.2.1) when errors indicate that a root link should be switched and when a communication attempt has terminated.

The above is accomplished via two subprocesses:

(1) Data/Status Processing

(2) Chain Status Processing

Data/Status Processing (4.1.2.2.1) collects data and status for each transaction to create Chain_Data_And_Status.

Chain Status Processing (4.1.2.2.2) records error information, computes when to automatically bypass transactions, controls Data/Status Processing (4.1.2.2.1), and notifies Queue Processing (4.1.2.1) of the Chain_Completion_Status.

Process Name: Data/Status Processing
Reference Number: 4.1.2.2.1
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Chain_Completion_Processing.-
Data_Status_Processing
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 3.2.2

Inputs:

- Input_Packet
- Transaction_Configuration_Data_Base
- Current_Network_Partition_Definition
- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- Chain_Identifier from Chain Status Processing (4.1.2.2.2)
- Transaction_Status from Chain Status Processing (4.1.2.2.2)
- Chain_Status from Chain Status Processing (4.1.2.2.2)

Outputs:

- Chain_Data_And_Status to I/O Request Processing (4.1.1)
- Packet_Status to Chain Status Processing (4.1.2.2.2)
- Chain_Transaction_Status to End Of Chain I/O Error Processing (4.1.2.2.2.2)

Notes: None

Description:

This process collects data and status for each transaction of a chain to create Chain_Data_And_Status. It is initiated by Chain Status Processing (4.1.2.2.2).

Each Input_Packet is made source congruent before it is used. Source congruency is performed based on the Current_Network_Partition_Definition such that, for each transaction, an Input_Packet is selected from the channel connected to the I/O Interface (4.1.3) that actually performed, or attempted to perform, the transaction. This process produces Packet_Status from Input_Packet.

Frame_Protocol_Error_Indicator and Transaction_Timeout_Indicator are copied directly from Input_Packet.Interface_Status.

Incorrect_Message_Length_Indicator is set if Input_Packet.Frame_Length does not match the expected value of Frame_Length in IO_Data_Base for the transaction.

Address_Mismatch_Indicator is set if Input_Packet.Network_Address does not match the expected value of Network_Address in IO_Data_Base for the transaction.

For Response Frames from DIUs, Encoded_Address_Indicator is set if the values of Input_Packet.Network_Address and the Encoded_Address (found in Input_Packet.Data) do not correspond.

Finally, the Residual_Bit_Count_Indicator is set if Input_Packet.Interface_Status.Residual_Bit_Count does not match the expected value of Residual_Bit_Count in IO_Data_Base for the transaction.

A Transaction_Status is received for each Packet_Status. The data from each Input_Packet is combined with the Bypass_Indicator and Comfault_Indicator indicated by Transaction_Status and collected in Chain_Data_And_Status.

This process uses Chain_Status plus error information about each transaction in the chain to form Chain_Transaction_Status. Chain_Transaction_Status indicates whether any transactions were performed; and if there were, whether there were no errors in the chain, at least one error, or errors in all transaction in the chain.

Process Name: Chain Status Processing
Reference Number: 4.1.2.2.2
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Chain_Completion_Processing.-
Chain_Status_Processing
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 3.1.4.4.2, 3.2.2, 5.1.1.1, and 5.1.1.2

Inputs:

- Current_Network_Partition_Definition
- End_Of_Chain_Status
- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- IO_Data_Base.Limits_Definition
- Transaction_Configuration_Data_Base
- Chain_Timeout_Value from Queue Processing (4.1.2.1)
- Packet_Status from Data/Status Processing (4.1.2.2.1)
- Chain_Transaction_Status from Data/Status Processing (4.1.2.2.1)

Outputs:

- GPC_Subscriber_IO_Error_Log
- Transaction_Configuration_Data_Base
- Chain_Completion_Status to Queue Processing (4.1.2.1)
- Chain_Identifier to Data/Status Processing (4.1.2.2.1)
- Transaction_Status to Data/Status Processing (4.1.2.2.1)

Notes: None

Description:

This process controls Data/Status Processing (4.1.2.2.1), records error information for a chain of transactions, computes when to automatically bypass transactions, and notifies Queue Processing (4.1.2.1) of the Chain_Completion_Status.

The above is implemented via two subprocesses:

(1) End of Chain Monitor

(2) End of Chain I/O Error Processing

End of Chain Monitor (4.1.2.2.2.1) monitors End_Of_Chain_Status to determine when a chain of transactions has been completed and assures that this data is source congruent.

End of Chain I/O Error Processing (4.1.2.2.2.2) implements the remainder of this process when prompted by End of Chain Monitor (4.1.2.2.2.1).

Process Name End Of Chain Monitor
Reference Number: 4.1.2.2.2.1
Identifier: IO_System_Services.-
IO_User_Communication_Services.
Chain_Processing.Chain_Completion_Processing.-
Chain_Status_Processing.End_Of_Chain_Monitor
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 3.2.2

Inputs:

- Chain_Timeout_Value from Queue Processing (4.1.2.1)
- End_Of_Chain_Status from I/O Interface (4.1.3)
- Current_Network_Partition_Definition

Outputs:

- Completion_Status to End Of Chain I/O Error Processing (4.1.2.2.2.2)
- Chain_Status to Data/Status Processing (4.1.2.2.1)

Notes: None

Description:

This process generates a value for Completion_Status based on the reception of End_Of_Chain_Status and the state of an internal timer. The timer is started upon the reception of Chain_Timeout_Value.

End_Of_Chain_Status is made congruent when it is received. If the chain was executed on an I/O network that currently has more than one partition, the End_Of_Chain_Status values for each partition are used to initialize Completion_Status. The congruent version of End_Of_Chain_Status is output as Chain_Status.

If Chain_Complete occurs while the internal timer is running, the timer is turned off and Completion_Status is set to Chain_OK.

If Chain_Complete occurs when the internal timer is not running, Completion_Status is set to Unexpected_Chain_Complete. (This indicates the detection of a "bus busy" condition while this GPC was not using the network.)

If the internal timer expires before Chain_Complete is received, Completion_Status is set to Chain_Timeout.

In the case of a partitioned network, this process monitors all active partitions for completion.

This process also gets Chain_Identifier from Chain_Timeout_Value and outputs Chain_Identifier in Completion_Status.

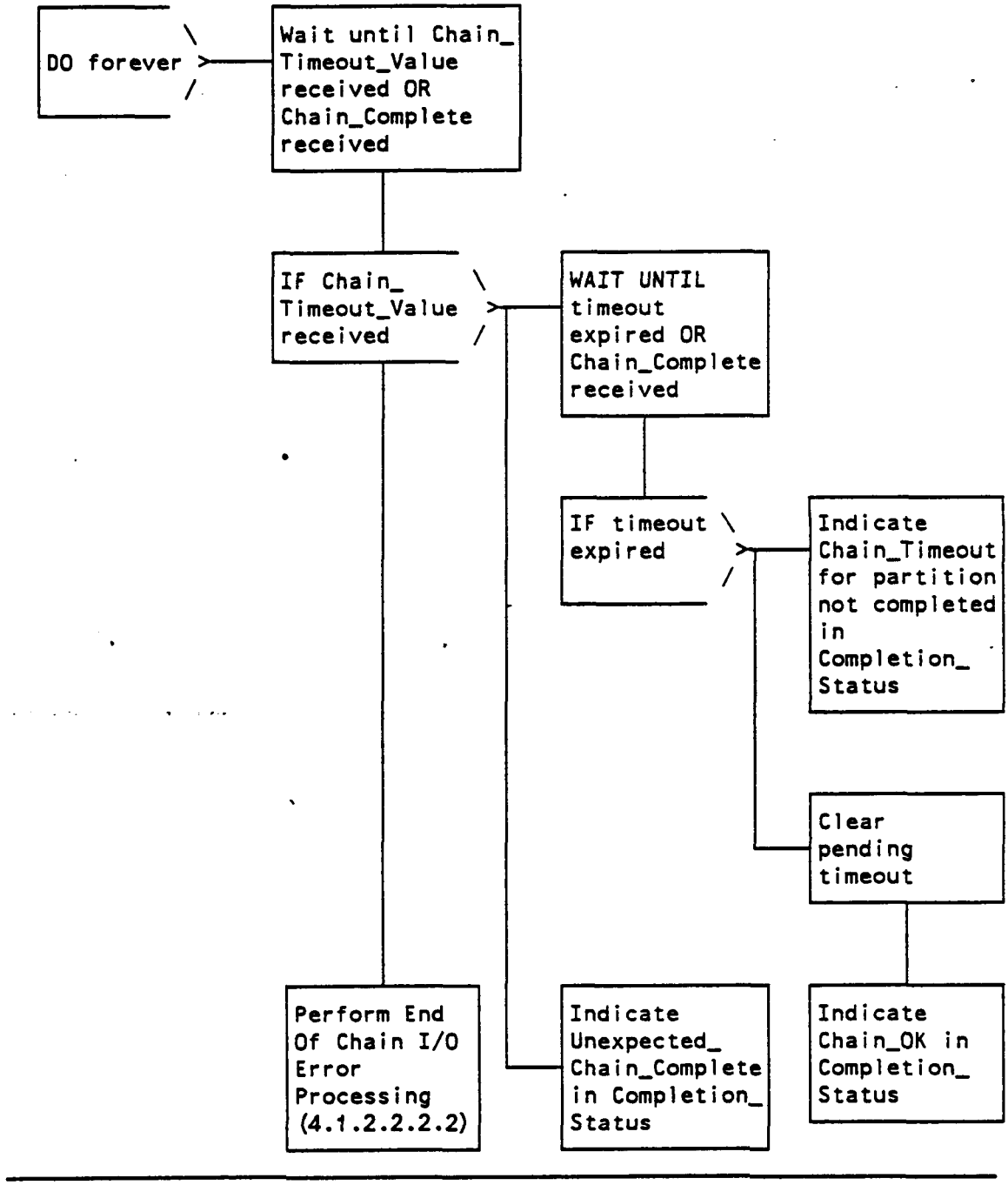


Figure 3-2. End Of Chain Monitor

Process Name End Of Chain I/O Error Processing
Reference Number: 4.1.2.2.2.2
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Chain_Completion_Processing.-
Chain_Status_Processing.
End_Of_Chain_IO_Error_Processing
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 3.1.4.4.2, 3.2.2, 5.1.1.1, and 5.1.1.2

Inputs:

- Completion_Status from End Of Chain Monitor (4.1.2.2.2.1)
- Packet_Status from Data/Status Processing (4.1.2.2.1)
- Transaction_Configuration_Data_Base
- IO_Data_Base.Limits_Definition
- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- Current_Network_Partition_Definition
- Chain_Transaction_Status from Data/Status Processing (4.1.2.2.1)

Outputs:

- Transaction_Status to Data/Status Processing (4.1.2.2.1)
- Chain_Identifier to Data/Status Processing (4.1.2.2.1)
- Chain_Completion_Status to Queue Processing (4.1.2.1)
- Update to Transaction_Configuration_Data_Base
- Update to GPC_Subscriber_IO_Error_Log

Notes: None

Description:

This process is initiated by the reception of Completion_Status.

Based on the values of Completion_Status and Chain_Transaction__Status, one of four cases is performed.

(1) When Completion_Status indicates an Unexpected_Chain_Complete, case 1 is performed.

(2) When Completion_Status indicates Chain_Timeout or Chain_Transaction_Status indicates No_Transactions_Performed, case 2 is performed.

(3) When Completion_Status indicates Chain_OK and Chain_Transaction_Status indicates that at least one transaction has an error, case 3 is performed.

(4) When Completion_Status indicates Chain_OK and Chain_Transaction_Status indicates that there were no errors among all the performed transactions, case 4 is performed.

Based on Completion_Status and Limits_Definition.Transaction_Error_Count_Limit, the process sets the value of Chain_Completion_Status to one of the following:

"Switch_Root_Link_And_Repeat_Chain" -- This is the case in which the GPC was unable to win a contention for the network for this running of the chain. On the assumption that this indicates a root link problem, the chain should be performed on another root link. In the event that the reason the GPC could not gain access to the network was a network-wide problem, such as a transmitter being stuck at one, the assumption is that the Network Manager will soon restore the network, and extra root link switches will be benign.

"Switch_Root_Link_And_Next_Chain" -- This is the case in which the GPC succeeded in winning a contention for the network, but was unable to perform any transaction in the chain successfully. A root link switch is performed in case the winning of the contention was allowed by a failure, such as the GPC's receiver for the active root link being stuck at zero. However, the chain is not retried automatically because it is possible that one or more of the errored transactions was an output that was actually performed, but there was an error on the expected acknowledgment. It might be dangerous to repeat the output, so the impetus for doing so is left to the caller.

"Next_Chain" -- This case comprises the situation in which no errors were detected, and the situation in which some transactions had errors and others were error free; that is, the situations in which at least one transaction was performed without error. In these situations it is most unlikely that a root link switch would accomplish anything.

This process obtains a value of Chain_Identifier as part of Completion_Status to indicate the identity of the chain being processed. It also outputs Chain_Identifier to Data/Status Processing (4.1.2.2.1).

Transaction_Status for each packet is set based on Completion_Status.Chain_Status and Packet_Status. This includes setting the Bypass_Indicator with the Transaction_Configuration_Data_Base.Bypass value for the transaction.

Transaction_Configuration_Data_Base components for each transaction, including Transaction_Error_Counter, Bypass, and Error_Process_Inhibit, are modified according to their previous val-

ues, Bypass_Enabled, Packet_Status, and Transaction_Error_Counter_Limit. Bypass_Enabled is specified in IO_Data_Base for each transaction; it indicates whether or not the transaction was defined with the "no transaction bypass" option.

GPC_Subscriber_IO_Error_Log is updated according to modifications made to Transaction_Configuration_Data_Base and the values of Chain_Completion_Status.

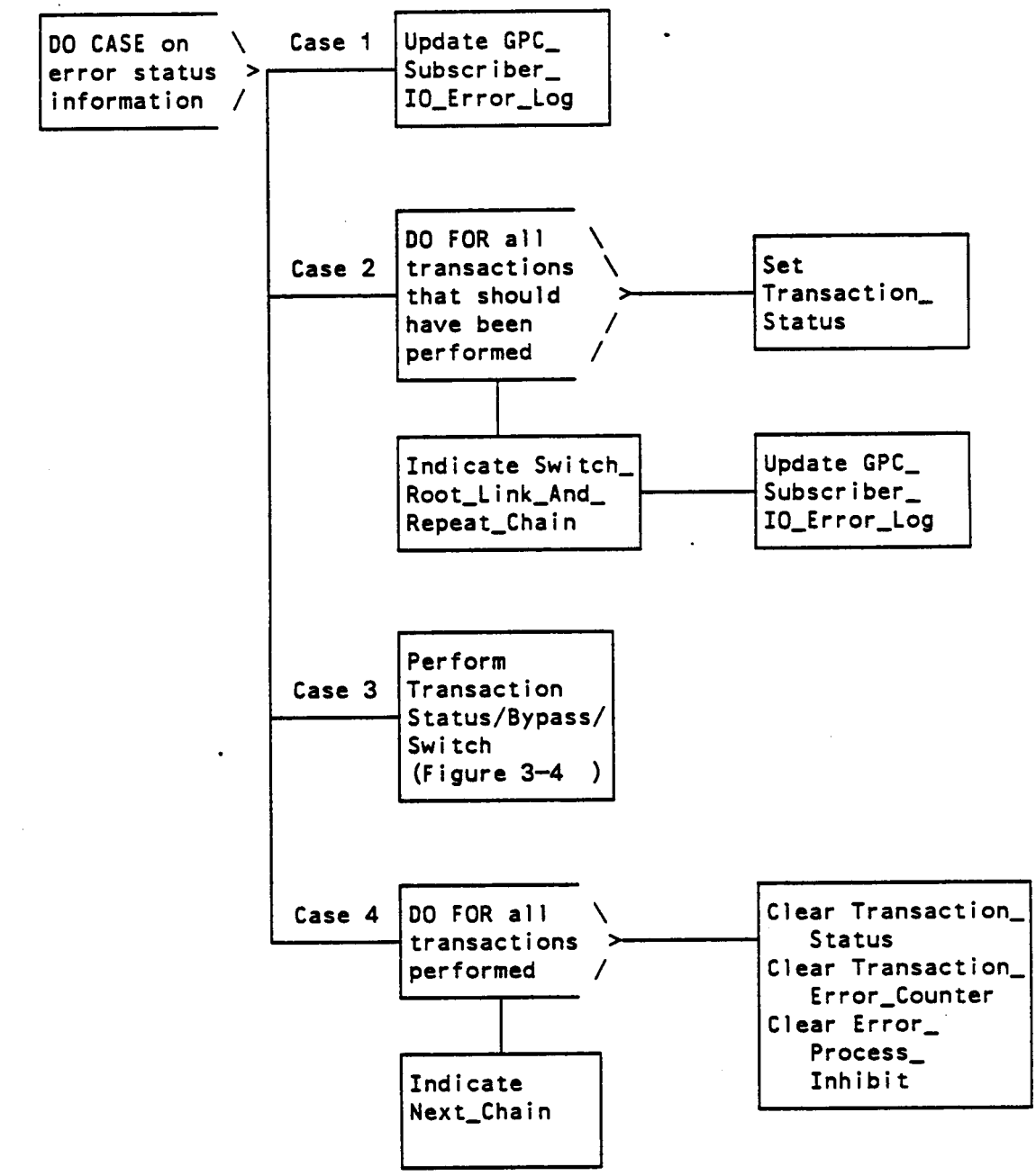


Figure 3-3. End Of Chain I/O Error Processing

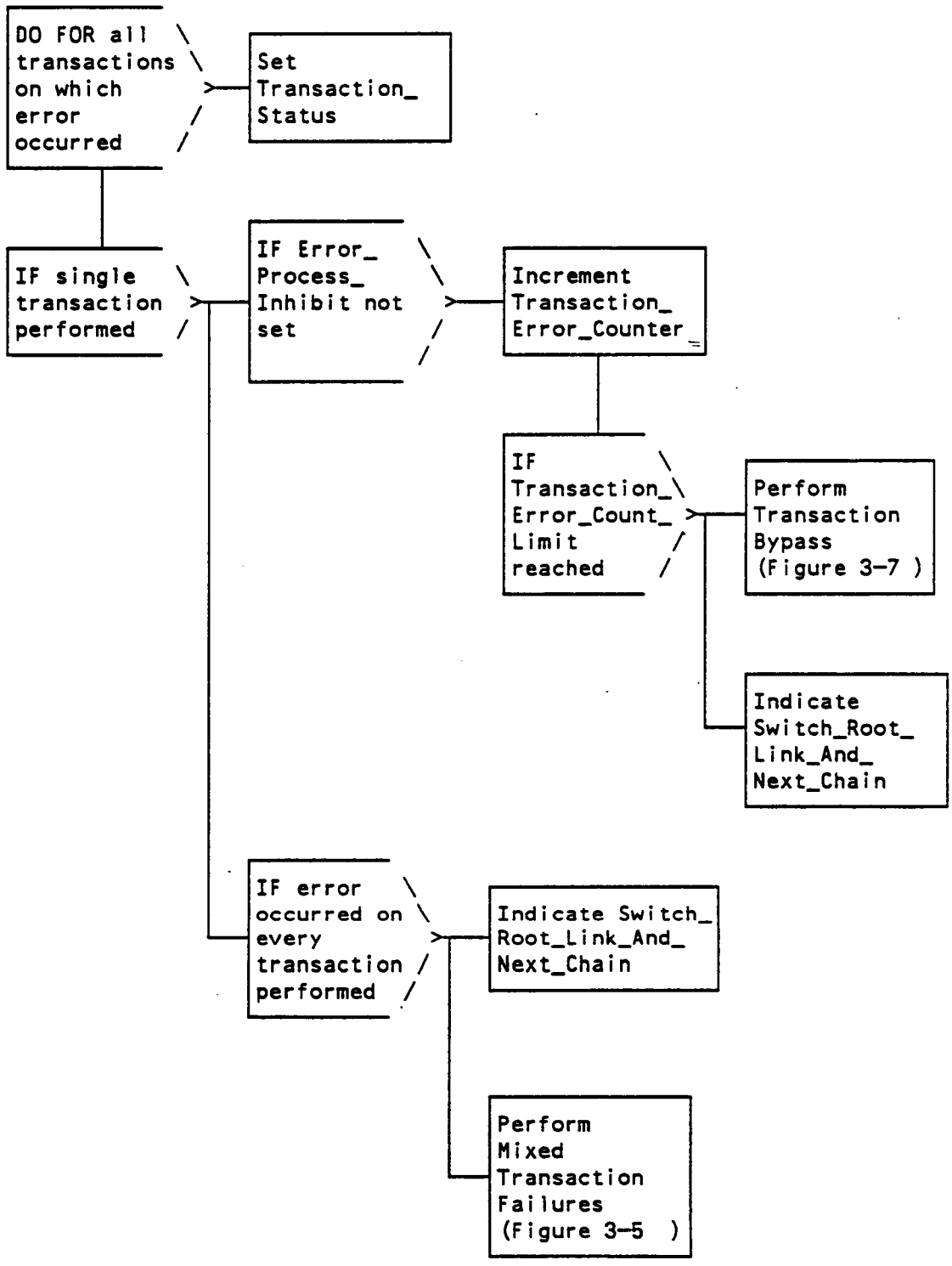


Figure 3-4. Transaction/Bypass/Switch

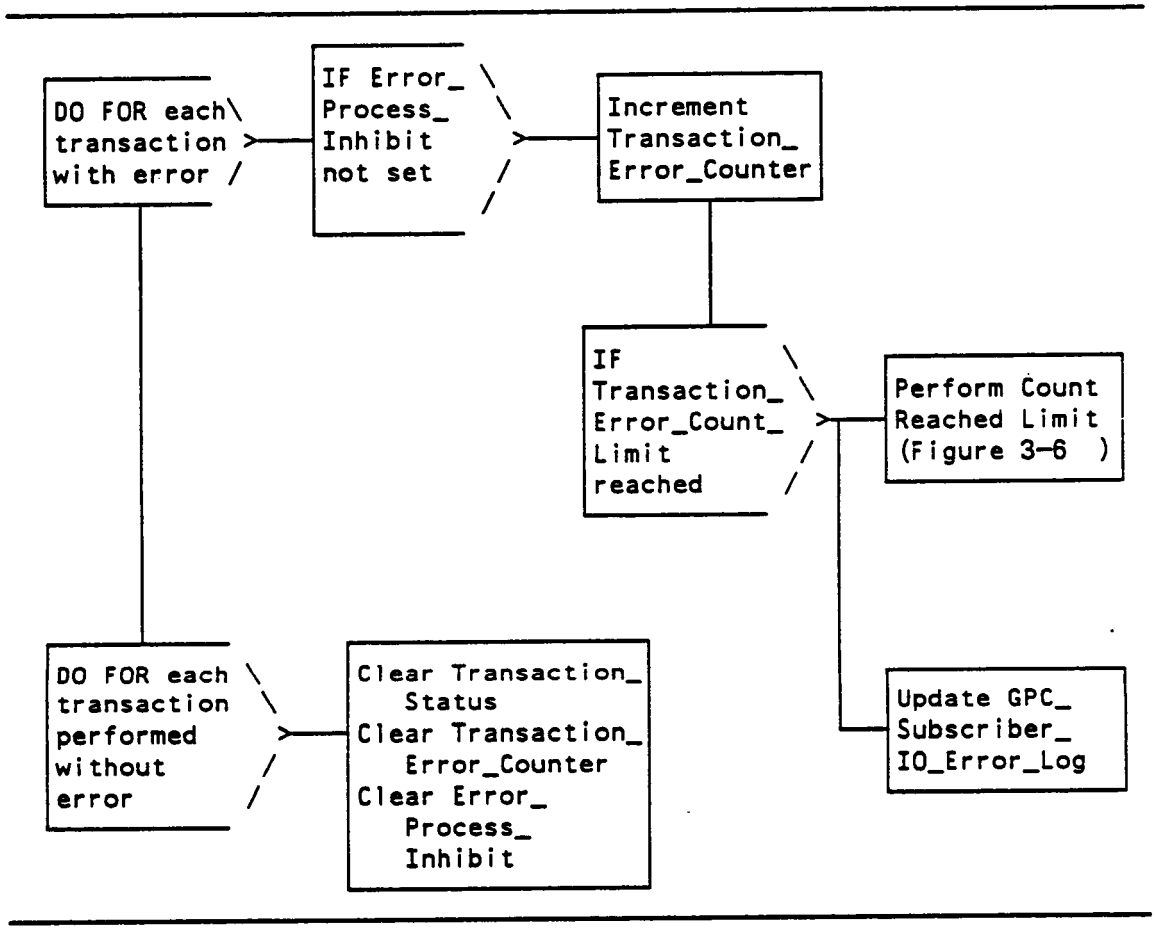


Figure 3-5. Mixed Transaction Failures

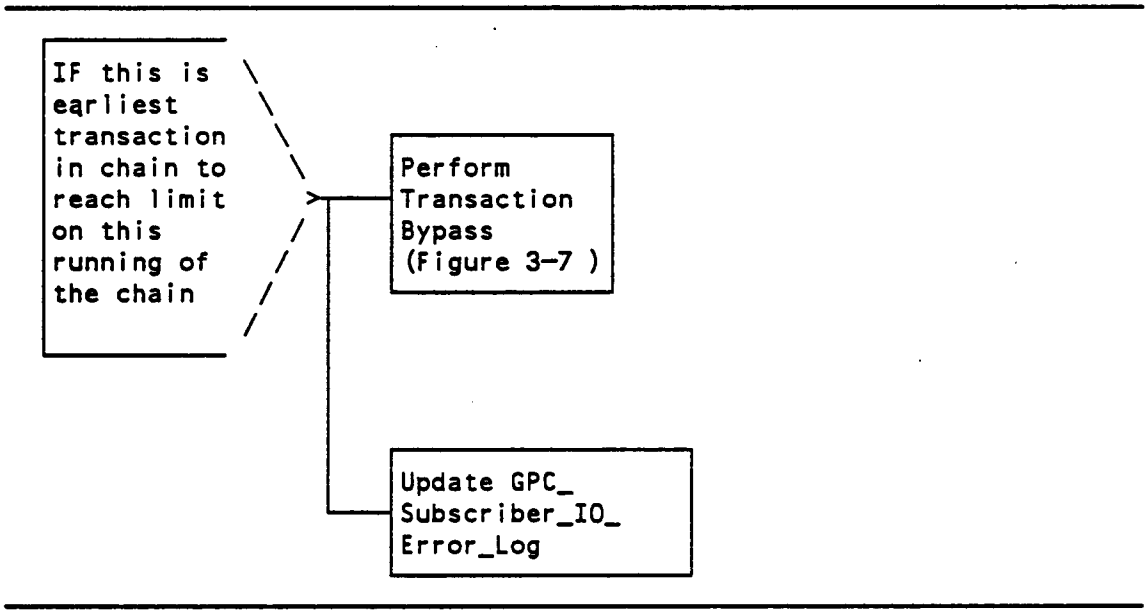


Figure 3-6. Count Reached Limit

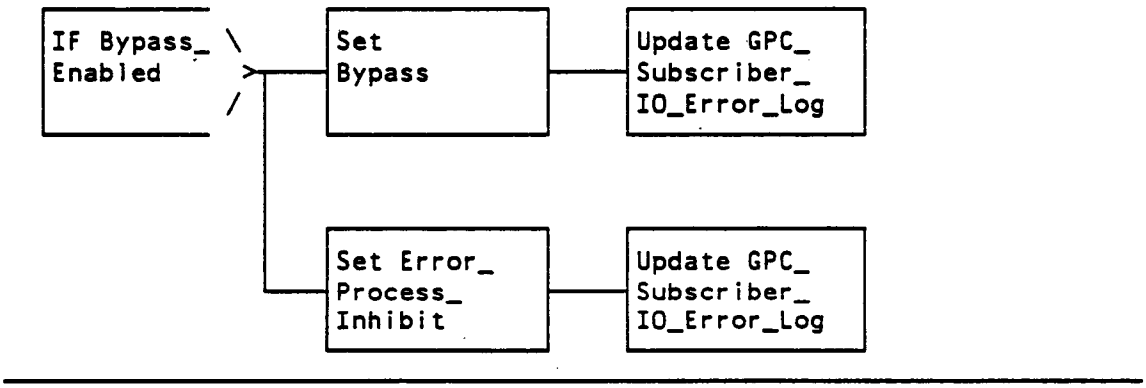


Figure 3-7. Transaction Bypass

Process Name: Chain Interface
Reference Number: 4.1.2.3
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Chain_Processing.-
Chain_Interface
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, Section 3.1.4.2

Inputs:

- Activate_Chain from Queue Processing (4.1.2.1)
- Root_Link_Command from Queue Processing (4.1.2.1)

Outputs:

- Chain_Initiation_Command to I/O Interface (4.1.3)

Notes: The implementation of this process must not be interrupted. The process monitors all inputs to detect when they change. This process may control other processes via interrupt signals

Description:

This process controls the access of Queue Processing (4.1.2.1) to I/O Interface (4.1.3).

It causes I/O Interface (4.1.3) to perform a chain of transactions by passing the Chain_Identifier, indicated by a new value in Activate_Chain, and the Root_Link_Identifiers, indicated by the last value received for Root_Link_Command, as Chain_Initiation_Command. The Root_Link_Identifiers indicate the active root link(s) for this I/O network.

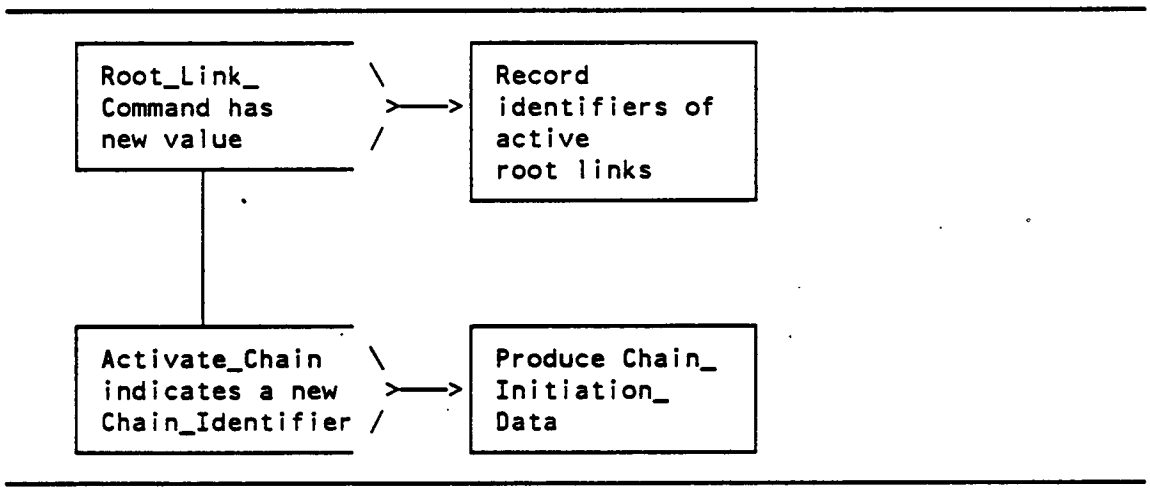


Figure 3-8. Chain Interface (4.1.2.3)

Process Name: I/O Interface
Reference Number: 4.1.3
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
IO_Interface
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 7

Inputs:

- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- Chain_Initiation_Command from Chain Processing (4.1.2)
- Current_Network_Partition_Definition from Chain Processing (4.1.2)
- Output_Packet from Chain Processing (4.1.2)
- Transaction_Configuration_Data_Base from Chain Processing (4.1.2)
- Response_Frame from DIU and Node (4.1.5)

Outputs:

- End_Of_Chain_Status to Chain Processing (4.1.2)
- Input_Packet to Chain Processing (4.1.2)
- Command_Frame to DIU and Node (4.1.5)

Notes: An instance of this process must exist for each I/O network connected to a GPC.

Description:

This process implements the performance of a chain of transactions to one or more DIUs or Nodes (4.1.5). The Chain_Identifier of the chain to be performed is specified by Chain_Initiation_Command. This input also specifies the root link processes that are to perform the chain. Data to be sent to each individual Node (4.1.5) or DIU is specified by an Output_Packet and communicated via Command_Frame. Data to be received from each Node (4.1.5) or DIU is received as a Response_Frame and stored as an Input_Packet. The completion status of a chain of transactions is specified in End_Of_Chain_Status. Other inputs are used by the various subprocesses to implement the above service.

The subprocesses include:

(1) Interface Initial Processing

(2) Interface Completion Processing

(3) Contention Processing

(4) Frame Transmitter

(5) Frame Receiver

Interface Initial Processing (4.1.3.1) sets up the other subprocesses and initiates the performance of a chain of transactions.

Interface Completion Processing (4.1.3.2) collects data and status from the performance of each transaction and the chain as a whole to produce the outputs End_Of_Chain_Status and Input_Packet.

Contention Processing (4.1.3.3) gains access to the I/O network for this process.

Frame Transmitter (4.1.3.4) transmits each Command_Frame.

Frame Receiver (4.1.3.5) receives each Response_Frame.

The last four subprocesses exist as a unit for each connection of a GPC to an I/O network and are known as root link processes. Each set is identified by a Root_Link_Identifier. These processes are performed in sequence as set up by Interface Initial Processing (4.1.3.1).

For the general case of chain performance, Contention Processing (4.1.3.3) (if used) is usually first. If completed successfully, it is followed by Frame Transmitter (4.1.3.4) and Frame Receiver (4.1.3.5) (if needed) for each transaction. Interface Completion Processing (4.1.3.2) is invoked in parallel with each Frame Receiver (4.1.3.4) invocation and may be invoked after the other subprocesses as well. It should be invoked at least once to produce End_Of_Chain_Status to indicate the outcome of Contention Processing (4.1.3.3).

Process Name: Interface Initial Processing
Reference Number: 4.1.3.1
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
IO_Interface.-
Interface_Initial_Processing
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 7

Inputs:

- Current_Network_Partition_Definition
- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- Output_Packet
- Transaction_Configuration_Data_Base
- Chain_Initiation_Command from Chain Processing (4.1.2)

Outputs:

- End_Of_Chain_Status_Identifier to Interface Completion Processing (4.1.3.2)
- Command_Frame_Status to Interface Completion Processing (4.1.3.2)
- Contention_Data to Contention Processing (4.1.3.3)
- HDLC_Program to Frame Transmitter (4.1.3.4)
- Output_Packet_Identifier to Frame Transmitter (4.1.3.4)
- Receiver_State to Frame Receiver (4.1.3.5)

Notes: None.

Description:

This process sets up the other I/O Interface (4.1.3) subprocesses to perform a chain of transactions to Nodes (4.1.5) or DIUs and initiates the performance of the chain.

The above is accomplished via the following subprocesses:

(1) Interface Chain Setup

(2) Interface Transaction Setup

Interface Chain Setup (4.1.3.1.1) sets up the root link processes for the chain of transactions in general. It also decides whether or not to set up the performance for each transaction in the chain.

Interface Transaction Setup (4.1.3.1.2) sets up the root link processes Frame Transmitter (4.1.3.4), Frame Receiver (4.1.3.5), and Interface Completion Processing (4.1.3.2) for each individual transaction, as directed by Interface Chain Setup (4.1.3.1.1).

Process Name: Interface Chain Setup
Reference Number: 4.1.3.1.1
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
IO_Interface.-
Interface_Initial_Processing.-
Interface_Chain_Setup
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 7

Inputs:

- Current_Network_Partition_Definition
- IO_Data_Base.IO_Request_Definition.IO_Chain_Definition
- Output_Packet
- Transaction_Configuration_Data_Base
- Chain_Initiation_Command from Chain Processing (4.1.2)

Outputs:

- Interface_Transaction_Data to Interface Transaction Setup (4.1.3.1.2)
- End_Of_Chain_Status_Identifier to Interface Completion Processing (4.1.3.2)
- Contention_Data to Contention Processing (4.1.3.3)
- HDLC_Program to Frame Transmitter (4.1.3.4)
- Receiver_State to Frame Receiver (4.1.3.5)

Notes: None

Description:

This process sets up the root link processes to perform the chain of transactions, decides whether or not to set up each transaction in the chain, and directs Interface Transaction Setup (4.1.3.1.2) appropriately. It then initiates the performance of the chain.

Contention_Data communicates the Contention_Priority to be used by Contention Processing (4.1.3.3) and the Contention_Limit, the number of times the contention should be attempted before it is considered failed.

HDLC_Program sets up Frame Transmitter (4.1.3.4) to communicate Command_Frames for Nodes (4.1.5) or Command_Frames for DIUs.

Receiver_State sets up Frame Receiver (4.1.3.5) to start or stop accepting Response_Frames. It is set up to start receiving frames before the first transaction is performed and to stop receiving frames after the last transaction has been completed.

Interface_Transaction_Data provides the data to Interface Transaction Setup (4.1.3.1.2) for each individual transaction. It communicates the identity of the Output_Packet used to initiate the transaction, the identity of the Input_Packet if a response frame is expected, the identity of the root link(s) to be set up, and either a Timeout_Value for determining when a transaction has failed due to the failure of a Response_Frame or a Time_Pad for determining how long to delay processing within a root link. These data items are indicated for each transaction by the IO_Chain_Definition specified by Chain_Initiation_Command. The IO_Chain_Definition also identifies each transaction within the chain.

The decision concerning each transaction is determined as follows:

- Processing for a particular transaction in the appropriate root link processes should be rearranged only if the conditions dictating the arrangement have changed since the last time they were examined. This rearrangement is performed by Interface Transaction Setup (4.1.3.1.2).

- Processing for this transaction in the appropriate root link processes is arranged so that this transaction will be performed when the chain is performed if and only if:

  - The value of Bypass is "No" as indicated by the Transaction_Configuration_Data_Base,
  - The value of Select is "Yes" as indicated by the Transaction_Configuration_Data_Base, and
  - The Network_Address specified by the Output_Packet for the transaction is reachable from the root link. This is indicated by indexing the Current_Network_Partition_Definition with the Network_Address.

  Otherwise, processing for this transaction in the appropriate root link processes is arranged so that this transaction will not be performed when the chain is performed.

- The processing for this transaction in the appropriate root link processes is replaced by a delay if and only if:

  - The network is partitioned into more than one partition,
  - The value of Bypass is "Yes" as indicated by the Transaction_Configuration_Data_Base,
  - The value of Select is "Yes" as indicated by the Transaction_Configuration_Data_Base, and
  - The Network_Address specified by the Output_Packet for the transaction is reachable from the root link. This is indicated by indexing the Current_Network_Partition_Definition with the Network_Address.

If the transaction is to be performed, the pertinent transaction data, i.e., Output_Packet_Identifier, and, if needed, Input_Packet_Identifier and Timeout_Value, are included in Interface_Transaction_Data.

If a transaction is to be replaced by a delay, only the Time_Pad value is included in Interface_Transaction_Data.

If a transaction is not to be performed, no data need be transferred via Interface_Transaction_Data.

After setup has been completed, the performance of the chain is initiated only in the root link processes indicated by Chain_Initiation_Command (i.e., only the root links which are active for this particular I/O network).

Process Name: Interface Transaction Setup
Reference Number: 4.1.3.1.2
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
IO_Interface.-
Interface_Initial_Processing.-
Interface_Transaction_Setup
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 7

Inputs:

- Interface_Transaction_Data from Interface Chain Setup (4.1.3.1.1)

Outputs:

- Output_Packet_Identifier to Frame Transmitter (4.1.3.4)
- Command_Frame_Status to Interface Completion Processing (4.1.3.2)

Notes: None

Description:

As directed by Interface Chain Setup (4.1.3.1.1), this process arranges the processing for individual transactions in the appropriate root link processes (i.e., Frame Transmitter (4.1.3.4), Frame Receiver (4.1.3.5), and Interface Completion Processing (4.1.3.2)). There are three processing options for a transaction:

- Perform the transaction when the chain is performed,
- Perform a delay (Time_Pad) instead of performing the transaction when the chain is performed, and
- Skip the transaction when the chain is performed.

One option is chosen for a transaction based on the input from Interface_Transaction_Data.

If Interface_Transaction_Data includes an Output_Packet_Identifier, the processing for this transaction, in the root link processes specified by the Root_Link_Identifier(s) in the input, is set up so that the transaction will be performed when the chain is performed. Specifically, the Output_Packet_Identifier is set up for the appropriate Frame_Transmitters (4.1.3.4).

If the input includes a Timeout_Value and an Input_Packet_Identifier, these items are set up, via Command_Frame_Status, for the appropriate

Interface Completion Processing (4.1.3.2) processes specified by the Root_Link_Identifier(s) in the input.

If the input includes a Time_Pad value, the processing for this transaction, in the root link processes specified by the Root_Link_Identifier(s) in the input, is set up so that a delay will occur instead of the transaction when the chain is performed.

If the input does not include values for Output_Packet_Identifier or Time_Pad, the processing for this transaction, in the root link processes specified by the Root_Link_Identifier(s) in the input, is set up so that the transaction is skipped when the chain is performed.

Process Name: Interface Completion Processing
Reference Number: 4.1.3.2
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
IO_Interface.-
Interface_Completion_Processing
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 7

Inputs:

- Command_Frame_Status from Interface Initial Processing (4.1.3.1)
- End_Of_Chain_Status_Identifier from Interface Initial Processing (4.1.3.1)
- Contention_Status from Contention Processing (4.1.3.3)
- Response_Frame_Data_And_Status from Frame Receiver (4.1.3.5)

Outputs:

- End_Of_Chain_Status
- Input_Packet

Notes: This process, Frame Transmitter (4.1.3.4), Frame Receiver (4.1.3.5), and Contention Processing (4.1.3.3) form the root link processes. There is one set of these processes for each connection between an I/O network and a GPC.

Description:

This process produces the End_Of_Chain_Status and Input_Packets for the chain performed via this root link. It consists of the subprocesses:

(1) Interface Chain Completion

(2) Interface Response Frame Processing

Interface Chain Completion (4.1.3.2.1) produces the End_Of_Chain_Status for the chain performed via this root link.

Interface Response Frame Processing (4.1.3.2.2) produces Input_Packet(s), if appropriate, for the transactions performed via this root link.

Process Name: Interface Chain Completion
Reference Number: 4.1.3.2.1
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
IO_Interface.-
Interface_Completion_Processing.-
Interface_Chain_Completion
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 7

Inputs:

- End_Of_Chain_Status_Identifier from Interface Initial Processing (4.1.3.1)
- Contention_Status from Contention Processing (4.1.3.3)
- Interface_Transaction_Status from Interface Response Frame Processing (4.1.3.2.2)

Outputs:

- End_Of_Chain_Status to Chain Processing (4.1.2)

Notes: This process, Frame Transmitter (4.1.3.4), Frame Receiver (4.1.3.5), Contention Processing (4.1.3.3), and Interface Response Frame Processing (4.1.3.2.2) form the root link processes. There is one set of these processes for each connection between an I/O network and a GPC.

Description:

This process computes the End_Of_Chain_Status based on Contention_Status and the Interface_Transaction_Status for each transaction that was performed in the chain.

The Interface_Transaction_Status for the performed transactions are used to compute the End_Of_Chain_Status values for All_Transactions_Failed_Indicator and At_Least_One_Transaction_Failed_Indicator.

Chain_Not_Processed_Indicator in End_Of_Chain_Status is used to indicate whether or not any transactions in the chain were attempted. This is based on whether or not Contention Processing (4.1.3.3) succeeded, as indicated by Contention_Status. If this indicates that contention failed, it is assumed that no transactions were performed.

The Bus_Error indicator in Contention_Status is also used to set the Bus_Error indicator in End_Of_Chain_Status.

When all processing is completed, the Chain_Complete indicator in End_Of_Chain_Status is set to indicate that the chain of transactions has been completed.

Process Name: Interface Response Frame Processing
Reference Number: 4.1.3.2.2
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
IO_Interface.-
Interface_Completion_Processing.-
Interface_Response_Frame_Processing
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 7

Inputs:

- Response_Frame_Data_And_Status from Frame Receiver (4.1.3.5)
- Command_Frame_Status from Interface Command Frame Processing (4.1.3.2.1)

Outputs:

- Input_Packet
- Interface_Transaction_Status to Interface Initial Processing (4.1.3.1)

Notes: This process, Frame Transmitter (4.1.3.4), Frame Receiver (4.1.3.5), Contention Processing (4.1.3.3), and Interface Chain Completion (4.1.3.2.1) form the root link processes. There is one set of these processes for each connection between an I/O network and a GPC.

Description:

This process creates Input_Packets for transactions expecting responses and controls the timing between frame transmissions through the use of delays.

If Command_Frame_Status includes a Timeout_Value and an Input_Packet_Identifier, this process produces an Input_Packet.

Otherwise, Command_Frame_Status includes only a Time_Pad value. In this case, processing is delayed for the length of time specified by Time_Pad and no outputs are produced.

If an Input_Packet is to be produced, a timer is set. If the timer runs out before Response_Frame_Data_And_Status is received, Interface_Transaction_Status is set to indicate "Failed" and the Input_Packet indicated by Input_Packet_Identifier is updated to indicate a Response_Frame timeout.

Otherwise, Response_Frame_Data_And_Status is transformed into Input_Packet. If the status portion of this input indicates a legal frame, then Interface_Transaction_Status is set to indicate "Successful".

Process Name: Contention Processing
Reference Number: 4.1.3.3
Identifier: IO_System_Services.-
IO_User_Communication.IO_Interface.-
Contention_Processing
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 7

Inputs:

- Contention_Data from Interface Initial Processing (4.1.3.1)

Outputs:

- Contention_Status to Interface Completion Processing (4.1.3.2)

Notes: Instances of this process communicate via Contention_Signals transmitted across the I/O Network

This process, Frame Transmitter (4.1.3.4), Frame Receiver (4.1.3.5), and Interface Completion Processing (4.1.3.2) form the root link processes. There is one set of these processes for each connection between an I/O network and a GPC.

Description:

This process contends for the I/O network to provide exclusive use of the network by the GPC. Contention is based on the Contention_Priority specified by Contention_Data.

If the process has not won the contention for the network after Contention_Data.Maximum_Attempts, it returns a Contention_Status value of "Failed". Otherwise, when it wins the contention, it returns a Contention_Status value of "Success".

If the process detects the bus busy or stuck-on-high condition, this information is indicated in Contention_Status.

Process Name: Frame Transmitter
Reference Number: 4.1.3.4
Identifier: IO_System_Services.-
IO_User_Communication_Services.
IO_Interface.-
Frame_Transmitter
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 7

Inputs:

- Output_Packet
- HDLC_Program from Interface Initial Processing (4.1.3.1)
- Output_Packet_Identifier from Interface Initial Processing (4.1.3.1)

Outputs:

- Command_Frame to DIU and Node (4.1.5)

Notes: This process, Frame Receiver (4.1.3.5), Contention Processing (4.1.3.3), and Interface Completion Processing (4.1.3.2) form the root link processes. There is one set of these processes for each connection between an I/O network and a GPC.

Description:

This process creates a command frame based on Output_Packet and HDLC_Program.

HDLC_Program specifies how many residual bits should be appended to the end of Command_Frame data. This programming remains in effect until the process is reprogrammed by Interface Initial Processing (4.1.3.1).

Each Output_Packet received is converted into a Command_Frame to be passed to a DIU or a Node (4.1.5) process, based on Output_Packet.Address and the current HDLC programming. The transformation is a copy of the Output_Packet with the generation of Frame_Check_Sequence based on the bit string representing the Output_Packet. Constant value opening and closing flags are also added as part of the Command_Frame.

Process Name: Frame Receiver
Reference Number: 4.1.3.5
Identifier: IO_System_Services.-
IO_User_Communication_Services.
IO_Interface.-
Frame_Receiver
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 7

Inputs:

- Receiver_State from Interface Initial Processing (4.1.3.1)
- Response_Frame from DIU and Node (4.1.5)

Outputs:

- Response_Frame_Data_And_Status to Interface Completion Processing (4.1.3.2)

Notes: This process, Frame Transmitter (4.1.3.4), Contention Processing (4.1.3.3), and Interface Completion Processing (4.1.3.2) form the root link processes. There is one set of these processes for each connection between an I/O network and a GPC.

Description:

This process creates Response_Frame_Data_And_Status in response to receiving a Response_Frame.

A Receiver_State value of "On" indicates the process should transform any Response_Frame that it receives to Response_Frame_Data_And_Status. A value of "Off" indicates that it should ignore any Response_Frame received.

Response_Frame_Data_And_Status consists of a copy of Response_Frame.Address, .Control, .Data, and .Frame_Check_Sequence, and error indicators, byte count, and residual bits detected by the process.

Process Name: GPC I/O Network Manager Support
Reference Number: 4.1.4
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
GPC_IO_Network_Manager_Support
Build: Post 3

Requirements Reference: Proof of Concept System Functional Requirements. I/O Network System Services. CSDL-AIPS-84-138, Section 5.0

Inputs:

GPC_Subscriber_IO_Error_Log

Outputs:

GPC_Subscriber_IO_Error_Report

Notes: None.

Description:

This process provides a subset of the GPC's I/O error information to the manager of a particular network.

Process Name: Node
Reference Number: 4.1.5
Identifier: IO_System_Services.-
IO_User_Communication_Services.-
Node
Build: NA

Requirements Reference: AIPS POC System Design Specification, Network Node

Inputs:

- Command_Frame from I/O Interface (4.1.3)

Outputs:

- Response_Frame to I/O Interface (4.1.3)

Notes:

- Nodes will be implemented with hardware and microprogram, not software
- A node may take up to 512 microseconds to begin to reply, starting from the reception of the end of Command_Frame
- Sumcheck in Command_Frame is intended to protect against erroneous data transformations that might occur in portions of the GPC that are not redundant. The sumcheck mechanism must allow the recipient to detect any error caused by a single fault that occurred before CRC was applied.

Description:

A Node performs the following functions:

- Detects activity and errors for each port
- Enables and disables ports for network connectivity
- Produces Response_Frames that contain information about detected activity and errors, and about the enable status of ports
- Accepts arbitrary inputs to a Message Buffer for future inclusion in Response_Frames

Activity and error detection are performed by the Node continuously. The detection of these conditions is latched in the Node's Status Register until the Status Register is cleared via a Command_Frame. The remaining functions are performed in response to Command_Frames.

On receipt of a Command_Frame addressed to the Node, the Node will check it for validity. If Command_Frame is valid, the node will perform one of the following sequences, as specified by the content of Command_Frame. Otherwise the Node will ignore the Command_Frame except to indicate the error in the Node's Status Register. Note that the Node produces a Response_Frame for every valid Command_Frame addressed to it.

- Replace the contents of the Node's Port Enable Register with the value included in Command_Frame and produce Response_Frame
- Update the Node's Message Buffer and produce Response_Frame
- Produce Response_Frame only

Each Command_Frame also controls the following:

- Whether the Response_Frame is to contain the Node's Status Register or Message Buffer
- Whether the Response_Frame is to be deliberately transmitted with a protocol error
- The number of residual bits to be included in Response_Frame
- The port or ports on which the consequent Response_Frame is to be transmitted
- Whether the Node's Status Register is to be cleared after transmission

When Response_Frame contains the Node's Status Register, the activity and error status transmitted is that before these indicators are cleared (if requested), and the port configuration status transmitted is that after being updated by Command_Frame (if requested).

The Command_Frame and Response_Frame formats are given in AIPS POC System Design Specification, Network Node.

A Command_Frame is honored by a Node if it meets the following criteria.

- Node_Address is the address of the node
- Encoded_Node_Address is valid
- Protocol checks (CRC, invalid frame, and abort) indicate legal protocol
- The number of bytes received is valid
- The number of bits after the last information byte, but before Frame_Check_Sequence, is valid (= 3)

- Sumcheck is valid

3.2 I/O Network Manager Processes

Process Name: IO_Network_Manager
Reference Number: 4.2.
Identifier: IO_System_Services.IO Network Manager
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, sections 4.0 and 5.0

Inputs:

- IO_Network_Manager_Command from System Manager (1.)
- Network_Definition from IO_Data_Base
- Manager_IO_Request_Data_And_Status from I/O User Communication Services (4.1)
- GPC_Subscriber_IO_Error_Report from I/O User Communication Services (4.1) of all GPC Subscribers to Network

Outputs:

- IO_Network_Status_Report to System Manager (1.)
- Manager_IO_Request_Parameter to I/O User Communication Services (4.1)
- Reconfiguration_Report to I/O User Communication Services (4.1) of all GPC Subscribers to Network

Notes: None.

Description:

Using the Network_Definition, this process will grow a fault tolerant network to allow GPC subscribers on the network to communicate serially with various I/O devices connected to the network. A network which provides I/O service to several GPCs is a regional network. Only one of these GPCs will run the IO Network Manager process for the regional network. However, this process is also capable of managing a local I/O network, that is one dedicated to the exclusive use of one GPC. Since such a network may be partitioned into several subnetworks, this process will be able to manage a partitioned network.

The network is grown or initialized by enabling various full duplex communication pathways through circuit switched nodes. Since not all possible pathways are enabled, the network has a set of spare links which allow it to be reconfigured in response to fault or damage events, rendering it resistant to such failures as a broken link, a transmitter or receiver stuck high or low, a babbling network element, or an element which responds to messages addressed to other elements.

The process periodically monitors the health of the I/O network by using the network to communicate with its nodes. This monitoring does not alter the configuration of a node. Rather a simple status read is requested of each node in the network. The node responds with its current configuration and an indication of whether or not it detected any errors since the last status read. I/O User Communication Services returns this data and the errors it logged while conducting the node transaction to the network manager in Manager_IO_Request_Data_And_Status. The ability to conduct error free transmissions of this type is evidence of a properly functioning communication link. Errors of either type are evidence of the existence of faults in the network.

Another source of error information comes from the GPC subscribers to the network who send GPC_Subscriber_IO_Error_Reports to this process. This process uses the error information it collects while monitoring the network and that sent by its GPC subscribers to identify the network elements responsible for the errors. It then reconfigures the network using spare links so as to isolate the failed component and maintain an active communication link to all functioning elements in the network.

Response to network failures is graduated in order to minimize the disruption of network activity by the repair process. Passive faults can be corrected most quickly while repair of a babbler may require regrowth of the network.

Whenever a reconfiguration has been completed, this process writes a Reconfiguration_Report to all GPC subscribers on the network. This will enable them to reinstate bypassed transactions. In this way I/O User Communication Services can resume I/O activity with devices which were temporarily out of service due to network problems.

This process communicates with nodes by sending them commands via I/O User Communication Services (4.1) which is sent a Manager_IO_Request_Parameter.IO_Service_Request for that purpose. The data field of the command frame contains a Sumcheck which the node uses in its error detection logic. This Sumcheck will be computed by all subprocesses sending messages to nodes prior to issuing a Manager_IO_Request_Parameter.IO_Service_Request.

When this process is configuring the network, either initially or in response to a failure, it will control the root link configuration of its host GPC. It will exercise this control by sending I/O User Communication Services a Manager_IO_Request_Parameter for that purpose. Another Manager_IO_Request_ Parameter will contain data regarding the current state of network partitioning which I/O User Communications Services needs to conduct I/O on a partitioned network.

This process will periodically report the status of the network nodes, either active or failed, to the global system manager so as to provide data for system FDIR. It will respond to IO_Network_Manager_Commands sent by the global system manager such as

a command to initialize the network as soon as the command is received.

In order to insure that spare links can be confidently called into service to reconfigure the network after a failure, a routine test of these spare links will be conducted. This test will also attempt to exercise links (i.e. ports) which have a failed status in Network_Status. Thus a link marked failed due to a transient error can have its status upgraded to null. Following the test, the network is returned to its pretest configuration. Since this test performs a contingency function only, it can be scheduled at a rate which is low enough to minimally interfere with higher priority processes.

C-2

Process Name: Control I/O Network
Reference Number: 4.2.1
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.0

Inputs:

- Network_Definition from IO_Data_Base
- Manager_IO_Request_Data_And_Status from I/O User Communication Services (4.1)
- IO_Network_Manager_Command from System Manager (1.)
- Network_Fault_Indicator from Monitor I/O Network (4.2.2)
- Network_Status
- Current_Network_Definition

Outputs:

- Manager_IO_Request_Parameter to I/O User Communication Services (4.1)
- Network_Initialization_Status to Network_Status
- Reconfiguration_Status to Network_Status
- Node_Status to Network_Status
- Current_Network_Definition

Notes: None

Description:

Using the Network Definition, this process will initialize or grow a network in response to an Initialize_Network_Command from the System Manager (1.). The growth of a network may also begin upon a system reset if the network is local. This process will control its host processor's root link configuration during this growth phase by sending a Manager_IO_Request_Parameter.Root_Link_Control_Request to I/O User Communication Services indicating the root link to be activated on the next transaction to a node of this network. Nodes will be sent commands to place their ports in a certain configuration by means of a Manager_IO_Request_Parameter.IO_Service_Request. Receipt of the configuration command will also cause the node to send back its current status by means of Manager_IO_ Request_Data_And_Status. The success or failure of the initialization process will be written to Network_Status as the Network_Initialization_Status.

After initializing the network, the Control I/O Network process is only required to resume activity in response to a network failure as indicated by the Network_Fault_Indicator from the Monitor I/O Network process (4.2.2). Using the Manager_IO_Request_Data_And_Status from

the chain on which the monitor detected errors, it must identify the location of the fault and reconfigure the network around the fault. During this attempt to reconfigure the network, it may be necessary to collect further information on the functioning of a particular communication link by additional reads of node status. This is done by sending the necessary Manager_IO_Request_Parameter to I/O User Communication Services. As with the initialization phase of this process, Control I/O Network will control the root link configuration of its processor to the network it is controlling. If the reconfiguration process is performed, this information will be written to Network_Status as Reconfiguration_Status. If any nodes are discovered to be failed and hence isolated from the active network, the failed status of these nodes is written to Network_Status in Node_Status. Since Network_Status is used by Report I/O Network Status (4.2.3) on a periodic basis, it will be necessary for Control I/O Network to lock Network_Status at the time it begins execution and to unlock it when the process is completed.

Finally, this process must respond to an indication that a partition has failed by repartitioning the network. The new partitioning of the network will be reported to I/O User Communication Services in a Manager_IO_Request_Parameter.Partition_Update_Request containing Current_Partition_Data.

Process Name: Control Network Definition
Reference Number: 4.2.1.1
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network.-
Control_Network_Definition
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.2

Inputs:

- Network_Definition from IO_Data_Base
- IO_Network_Manager_Command from System Manager (1.)
- Partition_Status from Network_Status

Outputs:

- Manager_IO_Request_Parameter to I/O User Communication Services (4.1) of host GPC
- Current_Network_Definition
- Node_Status to Network_Status

Notes: None

Description:

This process will update the Current_Network_Definition whenever necessary with a new definition read from the IO_Data_Base. After the Initialize I/O Network (4.2.1.2) process or the Maintain I/O Network (4.2.1.3) process has responded to this new definition, Control Network Definition will send to I/O User Communication Services of its host GPC a Manager_IO_Request_Parameter.Partition_Update_Request which reflects the current partitioning of the network in Current_Partition_Data.

There are three events which cause this process to update Current_Network_Definition:

(1) Receipt of an Initialize_Network command from the System Manager indicating that Current_Network_Definition must be initialized so that a network may be grown.

(2) Receipt of a Modify_Network_Definition command from the System Manager (1.). This command will be sent whenever a node is to be added to or removed from an existing network, thus changing the basic network definition. It will also be used to indicate that a failed node has been repaired and should be reconnected to the network.

(3) Partition_Status in Network_Status indicates a partition failure has occurred.

There are two events which cause this process to send Current_Partition_Data to I/O User Communications Services of its host GPC:

(1) Network_Initialization_Status indicates that a network has been successfully grown.

(2) Reconfiguration_Status indicates a change in a network configuration has been attempted.

In both cases it is important that the Current_Partition_Data be sent to I/O User Communications Services before the Reconfiguration_Report. In the case where Reconfiguration_Report announces a network initialization has been completed, correct partition information must be present for I/O activity to proceed. In the case where a reconfiguration report announces an attempt to repair a network, it causes transactions blocked by error indicators to be resumed. It is important that these reinstated transactions be sent through root links representing the current actual partitioning of the network.

This process will block other processes from reading Current_Network_Definition while it is obtaining a new value from IO_Data_Base to prevent use of a partially updated version.

Process Name: Handle Network Redefinition Events
Reference Number: 4.2.1.1.1
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network.-
Control_Network_Definition.-
Handle_Network_Redefinition_Events
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.2

Inputs:

- Partition_Status from Network_Status
- Network_Definition from IO_Data_Base
- IO_Network_Manager_Command from System Manager (1.)

Outputs:

- Send_Status to Network_Status
- Node_Status to Network_Status
- Current_Network_Definition

Notes: None

Description:

Upon reset (for a local network) or upon a Network_Initialization command (for a regional network) from the System Manager (1.), this process will lock the Current_Network_Definition. It will then initialize Current_Network_Definition from the IO_Data_Base. It will then send Network_Status a list of all nodes in the current network and mark their status null in Node_Status. It will indicate in Send_Status that Current_Partition_Data has not been sent. Finally, it will unlock Current_Network_Definition.

Upon receiving a Modify_Network_Definition command from the System Manager, this process will lock the Current_Network_Definition. It will then read a new definition from the IO_Data_Base and update Network_Status by adding a new node and marking its status null or by deleting a node as indicated by the command. It will indicate in Send_Status that Current_Partition_Data has not been sent. Finally, it will unlock Current_Network_Definition.

If Partition_Status indicates that a partition has failed, this process will obtain a new Current_Network_Definition from the IO_Data_Base for the particular partition failure shown. It is possible that no new definition will be issued. If a new definition is called for, it will first lock Current_Network_Definition. Then it will indicate in Send_Status that Current_Partition_Data has not been sent. It will then read into Current_Network_Definition the new net-

work definition data. Finally, it will unlock Current_Network_Definition.

Process Name: Send Current Partition Data
Reference Number: 4.2.1.1.2
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network.-
Control_Network_Definition.-
Send_Current_Partition_Data
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.2

Inputs:

- Reconfiguration_Status from Network_Status
- Network_Initialization_Status from Network_Status
- Send_Status from Network_Status
- Current_Network_Definition

Outputs:

- Manager_IO_Request_Parameter to I/O User Communication Services(4.1) of host GPC
- Send_Status to Network_Status

Notes: None

Description:

When Reconfiguration_Status or Network_Initialization_Status indicate that a change in the network has occurred, this process will determine the current partition state of the network from the Current_Network_Definition and send this Current_Partition_Data to I/O User Communication Services of its host GPC as part of a Manager_IO_Request_Parameter.Partition_Update_Request. It will then mark Send_Status as sent.

Process Name: Initialize I/O Network
Reference Number: 4.2.1.2
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network.-
Initialize_IO_Network
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.1

Inputs:

- Current_Network_Definition from Control Network Definition (4.2.1.1)
- IO_Network_Manager_Command from System Manager (1.)
- Manager_IO_Request_Data_And_Status from I/O User Communication Services (4.1)
- Regrow_Network_Request from Maintain I/O Network (4.2.1.3)

Outputs:

- Network_Configuration
- Node_Status to Network_Status
- Network_Initialization_Status to Network_Status
- Manager_IO_Request_Parameter to I/O User Communication Services (4.1)

Notes: None

Description:

Using the Current_Network_Definition, this process is used to initialize or regrow an I/O network. The algorithm used creates a maximally branching, minimum path from a single processor to each node in the network.

If the network is regional, the process will initialize that network upon receiving an Initialize_Network command from the System Manager (1.). If the network is local, the initialization is started upon a reset of the local GPC. However, the System Manager (1.) may also command the initialization of a local network. Furthermore, to reconfigure the network under worst case failure conditions, the Maintain I/O Network process (4.2.1.3) may find it necessary to regrow the entire network. It will issue a Regrow_Network_Request for this purpose. The Initialize Network process will be used to accomplish that reconfiguration.

This process will be able to initialize or regrow a partitioned network. However, each partition will be grown as an atomic unit rather than in parallel with the other partitions in the network. Thus partition #1 would be completely grown before the growth of partition #2

begins and so on. It must also be possible to regrow a single partition so that a failure in one partition can be repaired without disrupting communications on unfailed partitions.

While this process is running, it will coordinate control of its host GPC's root links to the network by sending the appropriate Manager_IO_Request_Parameter.Root_Link_Control_Request to I/O User Communication Services(4.1). This will provide initial assurance of a properly functioning root link to the network being managed. Furthermore, during the growth of a network, it is important that errors detected by I/O User Communication Services do not trigger a root link switch by that process. The growth algorithm is equipped to deal with these errors as indicated in Manager_IO_Request_Data_And_Status.

To add a new node to the active tree of the network, this process sends the node an appropriate Configuration_Command as a Manager_IO_Request_Parameter.IO_Service_Request. A properly functioning node will respond with its status which is returned to this process as Manager_IO_Request_Data_And_Status. If this transaction is completed without errors, it indicates that this node is now part of the active network. These transactions are conducted without contention for two reasons. In the first place, nonmanager GPC subscribers are connected to the network by the manager only after its initial growth is complete. Hence, during that growth period, there is no one on the network with whom the manager need contend. Secondly, during a network failure such as the presence of a babbling element on the network, contention mechanics may not be operable. Reinitialization of the network under failure conditions of this type is carried out to identify and isolate the babbler.

As each node is added to the network, the Network_Configuration is updated with the current configuration of each node on a port by port basis. Prior to beginning the initialization process, the status of each node in Node_Status is marked Null. Where an attempt to connect a node to the network is successful, the status of that node is changed to Active. If a node is found to respond to addresses other than its own, its status will be marked failed. Any nodes which still have a Null status after the network growth is completed will have their status changed to failed. Thus the failure of a single port of a node does not cause the entire node to be considered failed. Only after the growth process is complete will the identity of these unreachable nodes be apparent.

This process will set Network_Initialization_Status in Network_Status to its current state. During an initialization this state will be "Initialization In Progress: Final Status Pending". After an initialization attempt is completed three possible states could be recorded. These all begin with "Initialization Completed" followed by one of these modifiers: "Final Status Fully Successful" (when all nodes are active), "Final Status Partially Successful" (when at least one node is active), and "Final Status Failed" (when no nodes are active).

This process must coordinate the efforts of several subprocesses involved in initializing a network. When Handle Network Redefinition

Events (4.2.1.1.1) unlocks the Current_Network_Definition, this process will coordinate the growth of the network described therein with the following loop:

```
For each partition in the network
  Repeat
    Select A Root Link (4.2.1.2.1)
    Attempt to Grow to its Root Node (4.2.1.2.2)
  Until (a data link is established with a root node)
   or    (no more root nodes remain to be tried)
  If a data link is established with a root node
    then Complete Growth of the Partition (4.2.1.2.3)
    else indicate in Network_Status that the
         partition is failed
```

Process Name: Select Root Link
Reference Number: 4.2.1.2.1
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network.-
Initialize_IO_Network.Select_Root_Link
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.0

Inputs:

- Current_Network_Definition

Outputs:

- Current_Root_Link to Grow to Root Node (4.2.1.2.2)
- Manager_IO_Request_Parameter to I/O User Communication (4.1)

Notes: None

Description:

This process is called as the first step in the growth of a network partition. It obtains the next root link of the current partition, that is the one being grown, from Current_Network_Definition. It sends to I/O User Communication Services (4.1) a Manager_IO_Request_Parameter.Root_Link_Control_Request to obtain activation of the selected root link and to disable the root link switching capabilities of that process. Select Root Link then passes the address of the activated root link obtained from Current_Network_Definition to Grow to Root Node (4.2.1.2.2) in Current_Root_Link which will then begin its activity.

Process Name: Grow to Root Node
Reference Number: 4.2.1.2.2
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network.-
Initialize_IO_Network.Grow_To_Root_Node
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.0

Inputs:

- Current_Root_Link from Select Root Link (4.2.1.2.1)
- Manager_IO_Request_Data_And_Status from I/O User Communication Services (4.1)
- Current_Network_Definition

Outputs:

- Link_Status to Network_Status
- Node_Status to Network_Status
- Node_Configuration to Network_Configuration
- Manager_IO_Request_Parameter to I/O User Communication Services (4.1)
- Active_Root_Node to Complete Partition Growth (4.2.1.2.3)

Notes: None

Description:

This process looks up the Current_Root_Link in the Current_Network_Definition to obtain the address of the node it must enable. It sets up the configuration command to be sent to this root node. The configuration command will tell the node which of its five ports to enable and which to disable. When growing to a root node only the port attached to the root link is enabled. The other four are disabled. The port number to be enabled is obtained from the Current_Network_Definition. The process sends this configuration command as part of a Manager_IO_Request_Parameter.IO_Service_Request. The process also specifies that this transaction is performed without contention.

When Manager_IO_Request_Data_and_Status is ready, the process will resume. It will first check the error indicators therein to determine whether or not the transaction was in fact sent and if any transmission errors were detected during the transmission. If the transaction was conducted with no errors, the data will be checked to verify that the node has implemented the correct configuration. If no transmission errors were detected and the node configuration is correct, the Node_Status of this node is marked Active and the configuration of this node is recorded in Network_Configuration. To

accomplish the latter means that the Node_Configuration of this node will show the enabled port marked Inboard and the other ports marked Null. The Link_Status of the root link is also marked active. Since a successful data link to a root node has been established, Active_Root_Node can be assigned its address. Processing control can now pass to the Complete Partition Growth process (4.2.1.2.3) to which Active_Root_Node will be sent.

If error indicators are present in Manager_IO_Request_Data_And_Status or if the port configuration data sent back from the node does not match the configuration that was sent to it, the configuration command will be sent again. This is done to allow for the possibility that the error indicators were set by transient faults in the network and because failure to grow to a root node results in the failure of the entire partition when there is only one root link to the partition.

If the second try is successful, the data structures are updated and processing continues as described above. If the second try fails, then in Network_Configuration.Node_Configuration the status of the port which could not be enabled is marked failed and the Link_Status of this root link is marked failed. If another root link to that partition exists, processing control passes to Select A Root Link (4.2.1.2.1). However, if no root links remain to be tried, Partition_Status of this partition in Network_Status is marked failed. Root links which have a failed status are not permanently failed, but instead will be routinely retried during spare link testing. If they operate properly at that time, their status will be upgraded to Null.

Process Name: Complete Partition Growth
Reference Number: 4.2.1.2.3
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network.-
Initialize_IO_Network.Complete_Partition_Growth
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.1

Inputs:

- Current_Network_Definition
- Manager_IO_Request_Data_And_Status from I/O User Communication Services (4.1)
- Active_Root_Node from Grow To Root Node (4.2.1.2.2)

Outputs:

- Node_Configuration to Network_Configuration
- Link_Status to Network_Status
- Partition_Status to Network_Status
- Manager_IO_Request_Parameter to I/O User Communication Services (4.1)

Notes: None.

Description:

This process becomes active once Grow To Root Node (4.2.1.2.2) receives a consistent response from a root node of the current partition and sends the Active_Root_Node to this process. The root node address initializes a spawning node queue. The root node becomes the first spawning node. The growth algorithm then enters a loop which consults the tables in Current_Network_Definition to obtain the identities and addresses of the network elements adjacent to the node at the top of the spawning node queue. This topmost node is called the spawning node. As the ports of the spawning node are enabled, its adjacent network elements are brought into the active tree. Configuration commands are sent to the nodes via Manager_IO_Request_Parameter.IO_Service_Requests sent to I/O User Communication Services (4.1).

If the adjacent element is a GPC, the port of the spawning node facing that element is placed on a GPC subscriber list. If the adjacent element is a DIU, the port of the spawning node is placed on a DIU subscriber list. These ports will be enabled after the network growth is complete.

If the adjacent element is a node, an attempt is made to create a functional link to that node. If the attempt is successful, this node is placed at the end of the spawning queue. Creating a functional

link requires that a port of the spawning node and a port of the adjacent or target node be enabled; the spawning node is enabled first. Enabling each port is accomplished by sending a Manager_IO_Request_Parameter.IO_Service_Request with the correct configuration of the node to I/O User Communication Services (4.1) which in turn sends the configuration command to the node. I/O User Communication Services returns the response of the node to this command in Manager_IO_Request_Data_And_Status. The absence of error indicators on both transactions (to the spawning node and to the target node) is evidence of a properly functioning link. As each port is processed, Link_Status in Network_Status and Node_Configuration in Network_Configuration are updated. The Link_Status of a link may be active or failed. The Configuration_Status of a port may be inboard, outboard, null or failed.

If the adjacent element is a DIU, that port is enabled. If no errors are reported in Manager_IO_Request_Data_And_Status from this transaction, the port remains enabled. However, if errors are reported, indicating a babbling DIU, the port is returned to an inactive state. The final status of the port is recorded in Configuration_Status.

When all ports of the current spawning node have been processed, the spawning node is removed from the spawning queue and placed on the active node list. The next node in the queue becomes the current spawning node and the cycle repeats itself. When the spawning queue is empty, the partition growth of the node network is completed. In the case of a partitioned local network, the ports on the DIU subscriber list remain to be enabled. In the case of a regional network, the ports on both the DIU and GPC subscriber lists must be enabled.

The ports on the DIU subscriber list are enabled first. The correct configuration command is sent to each node on this list one at a time via a Manager_IO_Request_Parameter.IO_Service_Request. If no errors are reported in Manager_IO_Request_Data_And_Status from this transaction, the port remains enabled. If an error is reported, the port is returned to an inactive state. An error detected after enabling this port could be due to two causes: a failed network element (such as a a babbling DIU or a failure in the port adjacent to the DIU) or a DIU responding to a previously issued command from a GPC. The purpose of enabling the DIUs after the growth of the node network is completed is to allow enough time to elapse to ensure that a DIU would have completed any outstanding GPC commands. Thus any errors detected after enabling the port adjacent to a DIU are indications of a faulty network component. Node_Configuration and Link_Status are updated following each configuration attempt.

The ports adjacent to GPC subscribers on the subscriber list are also enabled one at a time. However, only the first of these nodes may be sent a message without contention. Once the network manager gives other GPCs access to the network, the manager must use the contention rules which govern access to a multiuser network. The Manager_IO_Request_Data_And_Status which is returned to this process after enabling the root node port of a GPC is ignored. Since a GPC

which is facing a port which is not enabled will not detect any network activity, it may be attempting to use the network at the time the port is enabled. This could result in errors being detected in the node's reply to its configuration command. To verify that the GPC is in fact not babbling, however, the manager must ask for a status read of that node with contention. If the transmission has errors, that port is returned to a null status. This phase of network growth is complete when all the ports on the subscriber list have been enabled and verified for proper functioning. Node_Configuration and Link_Status are updated following each verification transaction.

This growth algorithm generates the shortest path from the source processor to any node in the network. Furthermore, if a path exists to any node in a network, this algorithm ensures that it will be found and activated, even if the network is degraded by failures.

Two network failure modes are addressed and corrected by this algorithm, thus making it a useful backup tool for network maintenance when less drastic measures fail to isolate and remove a problem. The two failure modes are a babbling network element and a network node which talks out of turn, i.e. responds when another network node been addressed. The operation of the part of the algorithm which deals with these failures is described below.

When the process enables a port of the spawning node adjacent to a babbler, the babbler will interfere with the status report the spawning node sends following its reconfiguration. This will result in the reconfiguration of this port to a null state, thus isolating the babbler from the rest of the properly functioning network. The method works because the network links are full duplex and the reconfiguration command will reach the spawning node through the data line not corrupted by the babbler. If the spawning node itself is babbling from a spawning port, the target node will not respond to the corrupted message. Thus the target node will not be connected to the babbler.

After a new node appears to be successfully connected to the network, each node in the network is commanded to report its status, whether or not it is in the active tree. If an unconnected node (i.e. one which is not on either the spawning queue or the active node list) responds to this command, the most recently connected node is talking out of turn to this address. This newly added node must be disconnected from the active tree by setting the correct spawning node port to a null state. Furthermore, its status in Node_Status is marked failed, since the address decoding function of a node is a central function, independent of the port receiving the address. A previously connected node could also respond with errors. This means that either this node has recently failed or the most recently added node is talking out of turn. This last added node is then removed from the network as described above. The node or nodes which had errors on the previous test are again queried for status. If the error indicators are gone, it confirms the talker out of turn hypothesis, and the status of the removed node is set to failed. If not, it indicates that a failure has occurred during the growth process. In the former

case, the growth process is continued. In the latter case, the growth process must begin again from Select Root Link (4.2.1.2.1).

Once the network growth is completed, any nodes with a Null status are set to a failed status.

Process Name: Maintain I/O Network
Reference Number: 4.2.1.3
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network.-
Maintain_IO_Network
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.4

Inputs:

- Manager_IO_Request_Data_And_Status from I/O User Communication Services (4.1)
- Network_Fault_Indicator from Monitor I/O Network (4.2.2)
- Node_Configuration from Network_Configuration
- Current_Network_Definition
- Network_Status

Outputs:

- Manager_IO_Request_Parameter to I/O User Communication Services (4.1)
- Node_Configuration to Network_Configuration
- Regrow_Network_Request to Initialize I/O Network (4.2.1.2)
- Node_Status to Network_Status
- Link_Status to Network_Status
- Partition_Status to Network_Status

Notes: For Build 3 the response to a detected network fault will be to regrow the network.

Description:

This process has two related but nevertheless distinct functions. Its primary purpose is to restore the network to a fully operational state whenever the failure of some network component disrupts the proper functioning of the network. This is an aperiodic process which only becomes active in response to the detection of a fault in the network.

A second function of this process is to verify that the various error detection mechanisms in the nodes are operating properly. Furthermore , this second function must determine whether or not spare ports which the first function requires for repairs are operating properly. This process is periodic in nature. However, since it performs only a contingency function, it can be scheduled to run at a rate which does not interfere with the scheduling of higher priority processes.

Process Name: Repair Network Fault
Reference Number: 4.2.1.3.1
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network.-
Maintain_IO_Network.Repair_Network_Fault
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.4

Inputs:

- Manager_IO_Request_Data_And_Status from I/O User Communication Services (4.1)
- Network_Fault_Indicator from Monitor I/O Network (4.2.2)
- Node_Configuration from Network_Configuration
- Network_Initialization_Status from Network_Status
- Current_Network_Definition

Outputs:

- Manager_IO_Request_Parameter to I/O User Communication Services (4.1)
- Node_Configuration to Network_Configuration
- Predecessor_List to Network_Configuration
- Regrow_Network_Request to Initialize I/O Network (4.2.1.2)
- Node_Status to Network_Status
- Link_Status to Network_Status
- Partition_Status to Network_Status

Notes: For Build 3 the response to a detected network fault will be to regrow the network.

Description:

Each time Network_Initialization_Status indicates a new network has been grown, this process will compute a Predecessor_List for each node in the network whose status in Node_Status is marked Active. Predecessor_Lists are used in the identification of the source of a network failure. The algorithm for computing the Predecessor_Lists is as follows:

```
For each node in the network
 Include the Current Node on its own Predecessor_List
    Repeat
        Find the Inward Port of the Current Node
          in Node_Configuration
        Find the network element adjacent to this current node
                                through this Inward Port from
Current_Network_Definition
```

```
        If this element is a node
          then add this adjacent node to the Predecessor_List
        The adjacent node becomes the Current Node
    Until the adjacent network element is a GPC
```

Following the initialization of the Predecessor_Lists this process again becomes active upon receiving a Network_Fault_Indicator from Monitor I/O Network (4.2.2). The action taken by this process will depend upon the type of failure reported in the Network_Fault_Indicator.

If a node is transmitting valid data through a port which should be disabled, the node must be removed from the network. If this failed node is not removed, each time the manager asks for status from the node adjacent to this port, it would receive two valid commands to report its status. Only one response is expected. Once the first response is received, another node will be commanded to report its status. The second response of the node may interfere with the reply of the next node, making it appear that this next node has failed to respond correctly to a command. Once the failed node has been removed from the network, any nodes which had transmission errors reported against them should again be queried for status to determine whether or not the fault was due to the failure just corrected. Nodes now functioning properly can have their Network_Fault_Indicator cleared.

To remove a node from a network requires that nodes adjacent to its Outboard ports have their Inboard ports configured to be Null. Once this is accomplished, the node adjacent to its Inboard port shall have its corresponding Outboard port disabled as well. Next the Current_Network_Definition and the Network_Configuration are consulted to determine through which spare port the new link to these nodes will be made. Finally new links from the nodes adjacent to the chosen spare ports are established. The communication with a node is conducted by sending the appropriate Manager_IO_Request_Parameter.IO_Service_Request. The verification that the node has carried out the correct configuration command is contained in the Manager_IO_Request_Data_And_Status returned after each transaction with a node.

If the Network_Fault_Indicator shows that the contention mechanism has failed (e.g. a babbler keeps the network in an active condition for an excessive length of time preventing a contention from taking place, or a data line is stuck high making it impossible for any GPC to win a contention) the network will be regrown by sending to Initialize I/O Network (4.2.1.2) a Regrow_Network_Request.

If the Network_Fault_Indicator shows that one node in a partition has failed, an attempt will be made to reconnect that node to the network through one of its spare ports.

If the Network_Fault_Indicator shows that a subset of nodes on a partition has failed, the Predecessor_Lists of these nodes is compared to identify the site of the failure. The last entries in these lists should match until a failed node appears on each list. If they

do not match, it means that two or more faults have occurred. In this case, the partition will be regrown. If the entries do match however, the fault can be isolated by finding the first node from the end of the list which itself has been reported as having failed. An attempt to reconnect this node to the partition through one of its spare ports will be made. Following the success of this reconfiguration, the other failed nodes will be queried for status. If the reconfiguration does not bring all the failed nodes back into service, the partition is regrown.

Following a reconfiguration of a partition, the Predecessor_Lists are recomputed. Also Partition_Status and Reconfiguration_Status in Network_Status are updated to reflect the current state of the network or partition. Furthermore, following the reconfiguration of an individual node, Node_Configuration in Network_Configuration and Node_Status and Link_Status in Network_Status are also updated.

Process Name: Test Network Components
Reference Number: 4.2.1.3.2
Identifier: IO_System_Services.-
IO_Network_Manager.Control_IO_Network.-
Maintain_IO_Network.Test_Network_Components
Build: post 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.4

Inputs:

- Manager_IO_Request_Data_And_Status from I/O User Communication Services (4.1)
- Node_Configuration from Network_Configuration
- Current_Network_Definition
- Network_Status

Outputs:

- Manager_IO_Request_Parameter to I/O User Communication Services (4.1)
- Node_Status to Network_Status
- Link_Status to Network_Status

Notes: None

Description:

The function of this process is to verify that the various error detection mechanisms in the nodes are operating properly. Furthermore , this process must determine whether or not spare ports which are required for repairs are operating properly. This process is periodic in nature. However, since it performs only a contingency function, it can be scheduled to run at a rate which does not interfere with the scheduling of higher priority processes.

All null and failed links will be routinely tested. Null links are tested to determine if they can be safely brought into service to reconfigure around a failure. Failed links are tested to determine whether or not the initial assignment of failed status was due to a transient fault unrelated to the link itself. To accomplish this it may be necessary to keep track of a link's performance over time. Links which repeatedly change status probably have an intermittent failure of some kind and should be dropped from the spare/failed testing cycle. To test a link requires the identification of the nodes on either end of the link. One of these nodes is designated the spawning node and the other is designated the target node. The target node is first reconfigured so that all its ports are disabled. The configuration of the spawning node is modified so that the port adjacent to the target node is enabled while its other ports retain their original pretest status. The target node is then reconfigured

to enable the port adjacent to the spawning node. If the status returned to this process by the target node is error free, the link is operating properly. In this case, Link_Status and Node_Status are updated to show a null status for the link and ports involved. If the link is not operating properly, its Link_Status and Node_Status are declared failed. In either case the target node and the spawning node are returned to their pretest configurations but this time the spawning node is reconfigured first. The order in which target and spawning nodes are reconfigured prevents the possible formation of loops in the network.

Each of the error detection mechanisms in a node will be tested to determine that they are operating properly. Since the network management algorithms use this data to control the network, it is important that the data be valid. For example, a node can be commanded to transmit a frame from a given port which produces a CRC error in any port receiving the frame. Thus the ability to detect CRC errors in each port of a node can be verified by reading the status of the node being tested to clear its error indicators, commanding a node adjacent to the one being tested to send out a frame with bad CRC and then reading the status of the node being tested. If it has detected the error, it has passed the test.

Process Name: Monitor I/O Network
Reference Number: 4.2.2
Identifier: IO_System_Services.-
IO_Network_Manager.Monitor_IO_Network
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.3

Inputs:

- Manager_IO_Request_Data_And_Status from I/O User Communication Services (4.1)
- Network_Initialization_Status from Network_Status
- Node_Status from Network_Status
- Current_Network_Definition from Control I/O Network (4.2.1)
- GPC_Subscriber_IO_Error_Report from I/O User Communication Services (4.1) of all GPC Subscribers to the Network

Outputs:

- Manager_IO_Request_Parameter to I/O User Communication Services (4.1)
- Network_Fault_Indicator to Control I/O Network (4.2.1)

Notes: For the POC system the frequency at which this process runs is variable.

Description:

Monitor I/O Network becomes active when Network_Initialization_Status indicates an active network has been grown. This process gathers information on the health of the network from two sources: the network nodes and the GPC subscribers to the network. It uses this information to determine if any faults have occurred in the network. If any faults are detected, this information is passed on to the Control I/O Network process (4.2.1) in the Network_Fault_Indicator.

Process Name: Collect Network Status
Reference Number: 4.2.2.1
Identifier: IO_System_Services.-
IO_Network_Manager.Monitor_IO_Network.-
Collect_Network_Status
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.3

Inputs:

- Manager_IO_Request_Data_And_Status from I/O User Communication Services (4.1)
- Network_Initialization_Status from Network_Status
- Node_Status from Network_Status
- Current_Network_Definition from Control I/O Network (4.2.1)

Outputs:

- Manager_IO_Request_Parameter to I/O User Communication Services (4.1)
- Network_Fault_Indicator to Control I/O Network (4.2.1)

Notes: For the POC system the frequency at which this process runs is variable.

Description:

This process becomes active when Network_Initialization_Status indicates an active network has been grown. It will periodically command each node listed in the Current_Network_Definition which Network_Status does not declare failed to report its status by sending to I/O User Communication Services (4.1.) a Manager_IO_Request_Parameter.IO_Service_Request. These transactions are to be conducted on the network with contention.

Collect Network Status is the fault detection mechanism of the network manager. This detection mechanism operates in two stages.

When I/O User Communication Services transmits messages on the network to a node, it observes the success or failure of the communication and records those observations in the status field of Manager_IO_Request_Data_And_Status. This constitutes the first stage of fault detection and includes detection of the failure of a node to transmit a response to a command in a reasonable amount of time, the presence of transmission errors on the network during a response from a node, and the incorrect number of words in a response. In addition to detecting errors on transactions to individual nodes, in this stage the overall performance of the network is monitored for failures which impede the proper functioning of the contention sequence.

These failures include a babbler which is flooding the bus with meaningless signals and a data line which is holding the bus in a "stuck on one" condition. These errors are also reported to this process in the status field of Manager_IO_Request_Data_And_Status.

The second stage of error detection involves the processing of the information returned in the data field of Manager_IO_Request_Data_And_Status. This represents the node's perception of its current state. This data is processed to verify that the port configuration reported by the node is correct, that no activity is detected on a port not currently enabled, and that no valid frames have been received on a port not currently enabled. The detection of activity on a port which is not enabled is an indication that the node adjacent to that port is transmitting when it should be disabled. It is either babbling or transmitting valid data. The identification of which failure has occurred is made by observing whether or not the signals are valid (Valid_Frame_ Received is true). Furthermore, it may be possible to detect the existence of a recurring transient failure in the network by recording the node's detection of protocol errors on data flowing in the network for further analysis.

If any errors are detected by this process, those errors will be described in the Network_Fault_Indicator which is then sent to the Maintain I/O Network process (4.2.1.3).

Process Name: Collect Subscriber Status
Reference Number: 4.2.2.2
Identifier: IO_System_Services.-
IO_Network_Manager.Collect_Subscriber_Status
Build: post 3

Requirements Reference: POC System I/O Services Functional Requirements, section 4.2.1.3

Inputs:

- Network_Initialization_Status from Network_Status
- Node_Status from Network_Status
- Current_Network_Definition from Control I/O Network (4.2.1)
- GPC_Subscriber_IO_Error_Report from I/O User Communication Services (4.1) of all GPC Subscribers to the Network

Outputs:

- Network_Fault_Indicator to Control I/O Network (4.2.1)

Notes: For the POC system the frequency at which this process runs is variable.

Description:

This process will receive the GPC_Subscriber_IO_Error_Report. Errors reported here when no errors have manifested themselves in the periodic node status reads are evidence of transient faults in the network or of faults which the manager's use of the network does not trigger. This information is passed on to the Maintain I/O Network process (4.2.1.3) in the Network_Fault_Indicator.

Process Name: Report I/O Network Status
Reference Number: 4.2.3
Identifier: IO_System_Services.-
IO_Network_Manager.Report_IO_Network_Status
Build: 3 (Partial Implementation - See Subprocesses)

Requirements Reference: POC System I/O Services Functional Requirements, section 5.2.3

Inputs:

- Current_Network_Definition
- IO_Network_Manager_Command from System Manager (1.)
- Network_Initialization_Status from Network_Status
- Node_Status from Network_Status

Outputs:

- IO_Network_Status_Report to System Manager (1.)
- Reconfiguration_Report to I/O User Communication Services (4.1) of all GPC Subscribers to Network

Notes: None

Description:

If Network_Status.Initialization_Status indicates an initialization of the network or if Network_Status.Reconfiguration_Status indicates a reconfiguration of the network, send a Reconfiguration_Report to I/O User Communication Services (4.1) of all GPC Subscribers to network.

The IO_Network_Manager_Command is Send_Network_Status which will contain a parameter, Report_Rate, which will indicate whether the IO_Network_Status_Report is to be sent only upon a command from the System Manager (1.) or whether it will be sent periodically at a certain frequency until stopped by another Send_Network_Status command.

Process Name: Report To GPC Subscribers
Reference Number: 4.2.3.1
Identifier: IO_System_Services.-
IO_Network_Manager.Report_IO_Network_Status.-
Report_To_GPC_Subscribers
Build: 3

Requirements Reference: POC System I/O Services Functional Requirements, section 5.1.2.2

Inputs:

- Network_Identifier from Current_Network_Definition
- Send_Status from Network_Status
- GPC_Subscriber_List from Current_Network_Definition
- Network_Initialization_Status from Network_Status
- Reconfiguration_Status from Network_Status

Outputs:

- Reconfiguration_Report to I/O User Communication Services (4.1) of all GPC Subscribers to Network

Notes: None

Description:

This process will send a Reconfiguration_Report to the GPC Subscriber of a network whenever a reconfiguration of the network takes place and Send_Status indicates that Current_Partition_Data has been sent to I/O User Communications Services (4.1) of the host GPC. This includes the first time a network is configured or grown. Initialization_Status in Network_Status records the initial growth of a network. Reconfiguration_Status indicates that the network has been reconfigured to repair some failure or for some other reason such as a repartitioning of the network into subnetworks. These reports will contain the identifier of the network which has been reconfigured as indicated in Network_Identifier. The destination of these reports is obtained from the GPC_Subscriber_List.

Process Name: Report To System Manager
Reference Number: 4.2.3.2
Identifier: IO_System_Services.-
IO_Network_Manager.Report_IO_Network_Status.-
Report_To_System_Manager
Build: post 3

Requirements Reference: POC System I/O Services Functional Requirements, section 5.2.3.2

Inputs:

- IO_Network_Manager_Command from System Manager (1.)
- Network_Identifier from Current_Network_Definition
- Node_Status from Network_Status
- Network_Initialization_Status from Network_Status

Outputs:

- IO_Network_Status_Report to System Manager (1.)

Notes: None.

Description:

The IO_Network_Manager_Command, Send_Network_Status, will contain a parameter, Report_Rate, which indicates whether IO_Network_Status_Reports should be sent upon receipt of Send_Network_Status only or instead should be sent periodically at some frequency specified in Report_Rate.

The IO_Network_Status_Report generated by this process for the System Manager (1.) will contain information on the current state of the network as recorded in Network_Status. For example, if an attempt is made to initialize a network, this process will send a message to the System Manager (1.) based on the outcome of that attempt as logged in Network_Status.Initialization_Status by the Initialize I/O Network process (4.2.1.2). Network_Status may be locked by Control I/O Network (4.2.1). Thus Report to System Manager may have either to wait for Network_Status to be unlocked before sending its report or to send a report indicating that network status is pending.

Until the Initialize I/O Network process has grown or failed to grow a network, the status of all nodes in the network is null. After that process has run, nodes will have either an active or failed status logged in Network_Status.Node_Status by that process. Updates to Node_Status will be made by the Maintain I/O Network process (4.2.1.3) as necessary. Each IO_Network_Status_Report will indicate the current state of each node as logged in Node_Status. Finally, the IO_Network_Status_Report will include the address of the network which is the subject of the report as indicated in the Network_Identifier.

SECTION 4

DATA DICTIONARY

4.1 Introduction

This section contains a listing of every data item or data grouping identified on the data flow diagrams. Each entry includes a description of the data.

Note: Descriptions are made in the context of how the data item is used in process descriptions from Chapter 3 and data flow diagrams from Chapter 2.

The following conventions are used within data descriptions:

- Data items that are a composite of other data items are described in one of three ways:
  - A vertical list of data items, each preceded by a bullet, that make up the composite or
  - A list of data items connected by plus signs, e.g., Data + Status, or
  - An English description of the data item.
- Data items that are defined as a choice of one of a group of data items are described in one of three ways:
  - A vertical list, without bullets, of data items with a vertical bar symbol after each of the items excluding the last item,
  - A list of data items connected by vertical bars, e.g., Data | Data_Status | Status, or
  - An English description of the data item, e.g., "The item may be Data, Data_Status, or Status".
- Data items may be described as a choice of values. Values are represented by text within double quotes. (The name of a value may serve as its own description.)
- Data items that are defined as a collection of one or more data items (iterations of a data items) may be designated by placing the iterated data items in parenthesis, e.g., (Data_And_Status) to represent one or more elements made of Data_And_Status, and

(Data | Status) to represent a collection of one or more elements, any of which may be either a Data or a Status item.

- Where English description combined with the above syntactical symbols, it is either:

  - preceded by a double dash, "--", or
  - separated from the rest of the definition by a blank line.

4.2 Data

Activate_Chain

Chain_Identifier

Active_Root_Node

Holds the address of the root node from which a network partition will be grown. Its value is used to initialize the spawning node queue.

Address_Mismatch_Indicator

"Mismatch"
| "Match"

All_Transactions_Failed_Indicator

"All_Failed"
| "Some_Succeeded"

Application_Initialization_Request

- Service_Identifier
- Function_Identifier

At_Least_One_Transaction_Failed_Indicator

"All_Succeeded"
| "Some_Failed"

Bus_Error

This data item is an indication that the signal on the network is stuck on high, that contention was aborted, or the network is busy such that contention cannot occur.

Bypass

"Yes"
| "No"

Bypass_Clear

"Clear"
| "No-op"

Bypass_Clear_Queue_Element
(Chain_Identifier)

Bypass_Enabled

"Yes"
| "No"

Bypass_Indicator

"Executed"
| "Bypassed"

Chain_Complete

"Chain_Complete"
| "Not_Finished_Yet"

Chain_Completion_Status

"Next_Chain"
| "Switch_Root_Link_And_Repeat_Chain"
| "Switch_Root_Link_And_Next_Chain"

This data item indicates whether I/O errors encountered while performing a chain imply that root link switching should be performed or that the chain should be retried. The possible values are:

- Next_Chain indicates that root link switching should not be performed and the chain should not be retried.
- Switch_Root_Link_And_Repeat_Chain indicates that root link switching should be performed (if there is an alternative root link to switch to) and the chain should be retried after the switch.
- Switch_Root_Link_And_Next_Chain indicates that root link switching should be performed (if there is an alternative root link to switch to) but the chain should not be retried after the switch.

Chain_Data_And_Status

- Chain_Identifier
- (Transaction_Identifier + Data + Comfault_Indicator + Bypass_Indicator)

Chain_Identifier
-- identifies a chain within an I/O network

Chain_Initiation_Command

- Chain_Identifier
- (Root_Link_Identifier)

Chain_Initiation_Status

- Contention_Status
- Chain_Identifier

Chain_Not_Processed_Indicator

```
"Processed"
| "Not_Processed"
```

Chain_Queue

(Chain_Queue_Element)

-- This is a prioritized queue of Chain_Queue_Elements.

Chain_Queue_Element

```
Transaction_Queue_Element
| Root_Link_Queue_Element
| Bypass_Clear_Queue_Element
| Selection_Queue_Element
| Network_Partition_Queue_Element
```

Chain_Status

This data item has content identical to End_Of_Chain_Status.

Chain_Timeout_Value

This data item specifies how long an I/O chain is allowed to continue before being considered in error. The period being specified starts when the GPC performing the chain wins the contention. There is a unique value for every chain. This data item also indicates the Chain_Identifier for the chain.

Chain_Transaction_Status

This data item is a summary of errors in all transactions of the chain being processed. The following values are possible:

```
"No_Transaction_Performed"
| "No_Errors_In_Chain"
| "At_Least_One_Error_In_Chain"
| "Errors_In_All_Transactions"
```

Comfault_Indicator

```
"OK"
| "Faulted"
```

This data item indicates to the process that requested the transaction whether or not the data received due to the transaction is suspect, i.e., that an error occurred during the transaction or that the transaction was not performed.

Command_Frame

This data item specifies the operation or operations to be performed by a node or DIU, and the output data, if any, appropriate to those operations.

Figure C-2 on page C-4 shows the format of the packet portion (the portion between the Opening Flag and Closing Flag) of a Command_Frame.

Command_Frame_Status

Timeout_Value & Input_Packet_Identifier
| Time_Pad

Completion_Status

Indicates the Chain_Identifier for the completed chain and whether the determination of completion was based on its actually finishing, or on the detection of a timeout. The following values are possible.

- Chain_OK means that the chain in progress finished before the chain timeout limit was reached.
- Chain_Timeout indicates that the chain in progress was still in progress when the chain timeout limit was reached.
- Unexpected_Chain_Complete means that an end of chain condition was detected at a time when no chain was in progress on the network in question. This indicates the detection of a "bus busy" condition.

Contention_Data

- Contention_Priority
- Maximum_Attempts

Contention_Priority

-- symbol representing the relative level of priority with which to contend for the bus

Contention_Status

- "Contention_Won" | "Contention_Failed"
- Bus_Error

Current_Network_Definition

The version of Network_Definition currently being implemented.

Current_Network_Partition_Definition
(Partition_Identifier + (DIU_Address | Node_Address))

Current_Partition_Data
Data reflecting the current partitioning of a network by the network manager in accordance with the Current_Network_Definition. This data includes the number of partitions in a I/O network. It also includes the nodes, DIUs, and root links for each partition.

Current_Root_Link
The identifier of the active root link to a partition which is about to be grown.

Data
-- an array of one or more bits

DIU_Address
Network_Address -- for a DIU

Encoded_Address_Indicator

"Match"
| "Mismatch"

End_Of_Chain_Status

- Chain_Complete
- Chain_Not_Processed_Indicator
- All_Transactions_Failed_Indicator
- At_Least_One_Transaction_Failed_Indicator
- More_Than_One_Transaction_Performed_Indicator
- Bus_Error

Indicates whether the chain in progress has completed, and if so whether I/O errors were detected during chain execution.

(1) Chain_Complete indicates whether the chain in progress has completed.

(2) Error_Status indicates whether I/O errors prevented the GPC from winning a poll; whether there were I/O errors detected on all transactions in the chain, or on at least one transaction but not all; or whether response frames were expected from more than one DIU or node.

End_Of_Chain_Status_Identifier
This data item identifies an End_Of_Chain_Status data item.

Error_Process_Inhibit

"Yes"
| "No"

Frame_Check_Sequence
-- data computed by the interface to detect bit errors within a frame

Frame_Length
-- The combined length of the Network_Address, HDLC_Control_Byte, and Data fields produced by a Response_Frame for an Input_Packet

Frame_Protocol_Error_Indicator

"Yes"
| "No"

Function_Identifier
This data item uniquely identifies an application function.

GPC_Subscriber_Command
(Function_Identifier)

GPC_Subscriber_IO_Error_Log
This is a file that stores the history of I/O errors detected by local GPCs. Each log entry includes the type and time of the error.

GPC_Subscriber_IO_Error_Report
This is information from a GPC subscriber to the I/O Network Manager concerning the I/O errors the subscriber has experienced since the last such report.

GPC_Subscriber_List
The complete list of identifiers of GPC subscribers to an I/O network.

HDLC_Control_Byte
-- byte of information for HDLC control (not used)

HDLC_Program

"DIU_Communication"
| "Node_Communication"

Incorrect_Message_Length_Indicator

"Yes"
| "No"

Inhibit_Switching_Indicator

"Inhibit_Switching"
| "Allow_Switching"

Initialize_Network
: A command sent to the manager of a network by the System Manager (1.) requesting that a network be initialized or grown. It also contains an identifier of this network which selects the data describing this network in Network_Definition.

Input_Packet

- Interface_Status
- Frame_Length
- Network_Address
- HDLC_Control_Byte
- Data
- Frame_Check_Sequence

Input_Packet_Identifier
: Packet_Identifier -- for an Input_Packet

Input_Packet_Length
: This data item specifies the length of an Input_Packet

Interface_Chain_Timeout_Value
: This data item specifies the maximum period of time -- after a contention has been won and before a chain has completed -- before the chain is declared failed.

Interface_Status

- Transaction_Timeout_Indicator
- Frame_Protocol_Error_Indicator
- Residual_Bit_Count

Interface_Transaction_Data

- Timeout_Value
- Output_Packet_Identifier
- Input_Packet_Identifier

Interface_Transaction_Status

: "Successful"
: | "Failed"

IO_Chain_Definition

- Network_Identifier
- Chain_Identifier
- Chain_Timeout_Value
- Contention_Data
- End_Of_Chain_Status_Identifier
- HDLC_Program
- Request_Type
- (Transaction_Definition)

IO_Data_Base

- (IO_Request_Definition)
- (Limits_Definition)
- (Network_Definition)

This data item contains definitions for:

- I/O requests, including chain definitions and transaction definitions
- limits, including limits for switching as well as transaction error limits, and
- I/O networks, including nodes, links, and how they are connected.

Some components may be referenced individually or as a group. For example, IO_Chain_Definitions may be grouped according to usage by Application Function or according to the I/O network for which they are defined.

IO_Network_Command

GPC_Subscriber_Command
| IO_Network_Manager_Command

IO_Network_Manager_Command

A command from the System Manager (1.) to the process which will manage an I/O network. It may be one of the following:

Initialize_Network or
Modify_Network_Definition or
Send_Network_Status

IO_Network_Status_Report

This data is sent to the System Manager (1.) to indicate the status (active or failed) of each node in an I/O network. The status may also include current partition information.

IO_Request_Data

Data -- any number of bytes up to the maximum for a frame

IO_Request_Data_And_Status

- (Chain_Complete)
- (Data + Comfault_Indicator + Bypass_Indicator)

IO_Request_Definition

- IO_Request_Identifier
- Request_Type
- (IO_Chain_Definition)

IO_Request_Identifier
This data item uniquely identifies a specific I/O Request.

IO_Request_Parameter

```
IO_Service_Request
| Transaction_Selection_Request
| Application_Initialization_Request
```

IO_Request_Priority This value specifies the relative order in which Chain_Queue_Elements should be inserted into a Chain_Queue.

IO_Service_Request

- Service_Identifier
- IO_Request_Identifier
- (Data)

Limits_Definition
(Network_Identifier + Switching_Limit +
(Transaction_Identifier + Transaction_Error_Counter_Limit))

Link_Status
A table which has an entry for each link in the network. The status of a link may be one of the following:

"Active" -- the link connects two enabled ports and is transmitting data on the network.
"Null" -- the link is a spare part connecting two null ports and is not transmitting data on the network
"Failed" -- the link connects two ports, at least one of which has failed.

Local_IO_Status
(Time + Reason_For_Logging
+ Partition_Identifier
-- and one of the following

```
Root_Link_Status
| Root_Link_Status + Chain_Identifier
```

Manager_IO_Command

```
Manager_IO_Request_Parameter
| Reconfiguration_Report
```

Manager_IO_Data

```
Manager_IO_Request_Data_And_Status
| GPC_Subscriber_IO_Error_Report
```

Manager_IO_Request_Data_And_Status

- End_Of_Chain_Status

- (Data + Comfault_Indicator + Bypass_Indicator)
- Root_Link_Status

Three types of status are returned to the manager: End_Of_Chain_Status (status of the chain as a whole), the status of each transaction, and root link status (which root link was active for the chain). The Data is the information contained in the response frame of each transaction.

Manager_IO_Request_Parameter

```
IO_Service_Request -- a configuration command or a node
status read command
| Transaction_Selection_Request
| Root_Link_Control_Request
| Partition_Update_Request
```

Maximum_Attempts

-- number representing the maximum number of contention attempts to be made before declaring the contention failed

Modify_Network_Definition

The System Manager (1.) sends this command to an I/O Network Manager whenever the Network Definition is to be modified. The following changes can be made:

```
Node_Addition -- adds a node to the network
Node_Deletion -- removes a node from the network
Node_Repair -- indicates a failed node has been
-- repaired
```

In each case all changes to Network_Definition must be fully specified. For example, addition of a new node will require data to be entered in the link list, the node list on a port by port basis, the GPC list, etc.

More_Than_One_Transaction_Performed_Indicator

```
"One"
| "Many"
```

Network_Address

-- symbol representing an address on the I/O network

Network_Configuration

Information reflecting the enabled connections in the network.

- Node_Configuration
- Predecessor_List

Network_Definition

These are the tables which define the physical composition of the network. The information will include a node list,

a link list, and a GPC list. Furthermore, if the table applies to a local I/O network, it will also include information needed for network partitioning. The node list will define and identify for each port of that node, what its neighbor is (adjacent node and its identifier, DIU and its identifier, or GPC and its identifier). The link list will define for each link which two network elements it connects; one of the elements must be a node. The GPC list will define which link and node are connected to each GPC that is a network subscriber. The network partitioning information will include the number of partitions, the nodes in each partition, and the root links for each partition. Several versions of this partition data will be stored. The first version will assume no failed elements. Additional versions will provide similar information for repartitions of the network. Some alternate versions will be used to support repartitioning in response to network failures; others will support test objectives. A single I/O network may have multiple alternate definitions.

Network_Fault_Indicator
A list of all nodes which have errors reported in the status field of their Manager_IO_Request_Data_and_Status after a periodic status collection and all nodes which processing of the data collected from the nodes and GPC subscribers reveals to be failed.

Network_Identifier
-- symbol identifying a particular I/O network

Network_Initialization_Status
The current state of the network with respect to the activity of the Initialize Network process. The values of this data item are:

"Null" -- the Initialize Network process has not yet run
"Initialization in progress: Status pending" -- the process is currently attempting to grow a network
"Initialization Completed: Final Status Fully Successful" -- the process has connected all nodes in the network to the active tree
"Initialization Completed: Final Status Partially Successful" At least one node is in the active tree
"Initialization Completed: Final Status Failed" -- the process has not been able to connect any nodes in a network

Network_Partition_Queue_Element
Current_Partition_Data

Network_Status
A data structure which which contains current information about the state of various network elements and various

subsets of those elements, including the network itself. This information is distributed among the following fields:

- Link_Status
- Network_Initialization_Status
- Node_Status
- Partition_Status
- Reconfiguration_Status
- Send_Status

Node_Address
Network_Address -- for a Node (4.1.5)

Node_Configuration
A table which describes the configuration of each node in the network on a port by port basis. Each port may have the following status:

"Inboard" -- active and used as a spawning port when growing the network
"Outboard" -- active and used as a target port when growing the network
"Null" -- not currently enabled but believed to be functional and useful as a spare port
"Failed" -- not currently enabled but believed to be not functional and not useful as a spare port

Node_Status
The overall status of a node as well as its status on a port by port basis. If a node or port has a failed status, the diagnosed reason for the failure will be recorded. The status of a node or a port may be one of the following:

"Active" -- functioning normally in the network
"Null" -- not actively connected to network but not having been diagnosed as failed
"Failed" -- attempts to utilize the node or port result in errors being detected in the network.

Some reasons for declaring a node or a port failed are:

Passive failure
Babbling element
Talking out of turn
Transmitter stuck on

Output_Packet

- Network_Address
- HDLC_Control_Byte
- Data -- up to 128 bytes

Output_Packet_Identifier
Packet_Identifier -- for an Output_Packet

Output_Packet_Length
This data item specifies the length of an Output_Packet

Packet_Identifier
This data item uniquely identifies an Output_Packet or an Input_Packet.

Packet_Status

- Frame_Protocol_Error_Indicator
- Transaction_Timeout_Indicator
- Incorrect_Message_Length_Indicator
- Address_Mismatch_Indicator
- Encoded_Address_Indicator
- Residual_Bit_Count_Indicator

Partition_Identifier
-- symbol identifying a partition of a network. A network may be divided into one, two, or three partitions.

Partition_Status
The current state of a partition of a network. Each partition in a network has its own status. The value of this status may be:

"Active" -- all nodes on the partition are functioning properly
"Failed" -- at least one node in a partition is failed

Partition_Update_Request

- Service_Identifier
- Network_Identifier
- Current_Partition_Data

Predecessor_List
The ordered list of nodes through which a given node is connected to the host GPC of the I/O Network Manager. The first node on the list is the owner of the list itself. The last node is the node closest to the GPC hosting the I/O Network Manager process. Each node in the active tree has its own predecessor set and also appears as a member of that set.

Reason_For_Logging

"Root_Link_Switch"
| "All_Root_Links_On_Partition_Faulted"
| "Root_Link_Rotation_Limit_Reached"

Receiver_State

"On"
| "Off"

Reconfiguration_Report

This report indicates that the I/O Network Manager has conducted a reconfiguration operation. It contains an identifier of the network on which the operation has been performed. The manager sends such a report to the I/O User Communication Services of every GPC subscriber on the I/O network it is managing.

Reconfiguration_Status

The state of the network with respect to the Maintain I/O Network process. The status may be one of the following:

"Reconfiguration in progress: Outcome pending"
"Reconfiguration completed: Network Modified"
"Reconfiguration completed: No change in network"

Regrow_Network_Request

A request by the Maintain I/O Network process to reinitialize a network because a failure mode has been diagnosed which can only be repaired by completely regrowing the network.

Report_Rate

A value indicating the frequency at which the IO_Network_Status_Report is to be sent to the System Manager (1.). If the value is "0", then these reports are sent upon receipt of a Send_Network_Status command only.

Request_Type

"Manager"
| "IO_Request"

Residual_Bit_Count

This data item is the count of the number of residual bits (between 0 and 7) received from a Response_Frame.

Residual_Bit_Count_Indicator

"Match"
| "Mismatch"

Response_Frame

This data item is a node or DIU's reply to a command frame. Not all command frames evoke a response frame, but all response frames are triggered by a command frame. Figure C-3 on page C-5 shows the format of the packet portion (the portion between the Opening Flag and Closing Flag) of a Response_Frame.

Response_Frame_Data

- Network_Address -- same value used for the Command_Frame preceding the Response_Frame

- HDLC_Control_Byte
- Data
- Frame_Check_Sequence

Response_Frame_Data_And_Status

- Response_Frame_Status
- Response_Frame_Data

Response_Frame_Status

- Frame_Protocol_Error_Indicator
- Frame_Length
- Residual_Bit_Count

Retry_Limit
This data item indicates the number of times a the root links, as a group, should be switched during the attempted execution of one chain of transactions before the chain is declared failed.

Root_Link_Activation

"On"
| "Off"

Root_Link_Command
(Root_Link_Identifier + Root_Link_Activation)

Root_Link_Control_Request

- Service_Identifier
- Network_Identifier
- Inhibit_Switching_Indicator
- Root_Link_Identifier

Root_Link_Failure

"Good"
| "Bad"

Root_Link_Identifier
-- symbol identifying one of the I/O Interface (4.1.3) processes that connect the rest of I/O User Communication Services (4.1) to an I/O network.

Root_Link_Queue_Element
(Partition_Identifier)

Root_Link_Status

Root_Link_Command
| (Root_Link_Identifier + Root_Link_Failure)

Rotation_Log_Limit
This data item indicates the number of times the root links, as a group, should be switched before logging the event.

Selection

"Select"
| "Skip"

Selection_Default
Selection

This data item is a default value for Selection for a transaction. It is found in IO_Data_Base.-IO_Request_Definition.IO_Chain_Definition.Transaction_Definition.

Selection_Queue_Element
(Chain_Identifier + (Transaction_Identifier + Selection))

Send_Network_Status
A command from the System Manager (1.) to an I/O Network Manager requesting a report on the state of the network. It will contain the parameter Report_Rate to control the frequency of generation of these reports.

Send_Status
Indicates whether or not Current_Partition_Data has been sent to I/O User Communication Services (4.1). Its value may be one of the following:

"Sent"
"Not Sent"

Service_Identifier

"IO_Service_Request"
| "Transaction_Selection_Request"
| "Root_Link_Control_Request"
| "Partition_Update_Request"
| "Application_Initialization_Request"

Single_Link_Log_Limit
This data item indicates the number of times a root link may be switched before the switching is logged as a special event.

Switching_Limit

- Single_Link_Log_Limit
- Rotation_Log_Limit
- Retry_Limit

Time

-- System time as recorded by the local processing site.

Time_Pad

Timeout_Value -- for use when a transaction is bypassed within a partitioned I/O network.

Timeout_Value

This data item specifies the time period which may elapse before a Response_Frame is considered failed.

Transaction_Configuration_Data_Base

((Transaction_Identifier + Bypass + Bypass_Enabled + Error_Process_Inhibit + Transaction_Error_Counter + Selection) |
(Chain_Identifier + Bypass_Clear)

This data item contains the following information for each transaction.

(1) Bypass indicates whether the transaction has been bypassed.

(2) Bypass_Enabled indicates whether bypassing or error process inhibiting is to be performed in the event that the transaction's error counter reaches the limit.

(3) Error_Process_Inhibit indicates whether the transaction's error counter has reached the limit and that minimal I/O error processing should be performed in lieu of bypassing.

(4) Transaction_Error_Counter indicates the number of consecutive occurrences of the transaction on which an I/O error was detected, up to the point where transaction bypass or error process inhibit is performed.

(5) Selection indicates whether the transaction has been selected to be performed or skipped in this chain.

Transaction_Definition

- Transaction_Identifier
- Output_Packet_Length
- Output_Packet_Identifier
- Timeout_Value
- Time_Pad
- Input_Packet_Length
- Input_Packet_Identifier
- Selection_Default

Transaction_Error_Counter

This data item is an integer count ranging from zero to Transaction_Error_Counter_Limit

Transaction_Error_Counter_Limit
-- the number of consecutive occurrences of a transaction with a communications error that it takes to trigger bypassing or error process inhibiting

Transaction_Identifier
This data item identifies a transaction within a chain

Transaction_Queue_Element

- Chain_Complete
- Chain_Identifier
- IO_Request_Priority
- Root_Link_Status
- (IO_Request_Data)

Transaction_Selection_Request

- Service_Identifier
- IO_Request_Identifier
- (Chain_Identifier + (Transaction_Identifier))

Transaction_Status

- Comfault_Indicator
- Bypass_Indicator

Transaction_Timeout_Indicator

"Yes"
| "No"

Wait_Enqueue
-- indication to Local O.S. to enqueue a process on the local wait queue

Wait_Request

Wait_Enqueue
| Wait_Request_Dequeue

Wait_Request_Dequeue
-- indication to Local O.S. to release a process from the local wait queue

SECTION 5

PROCESS AND DATA LOCATION

The process and data locations specified herein are based on the FTP design as described in [4]. These major FTP elements are shown in Figure 5-1. This is a greatly simplified depiction of the FTP; only one replication of the redundancy is shown.

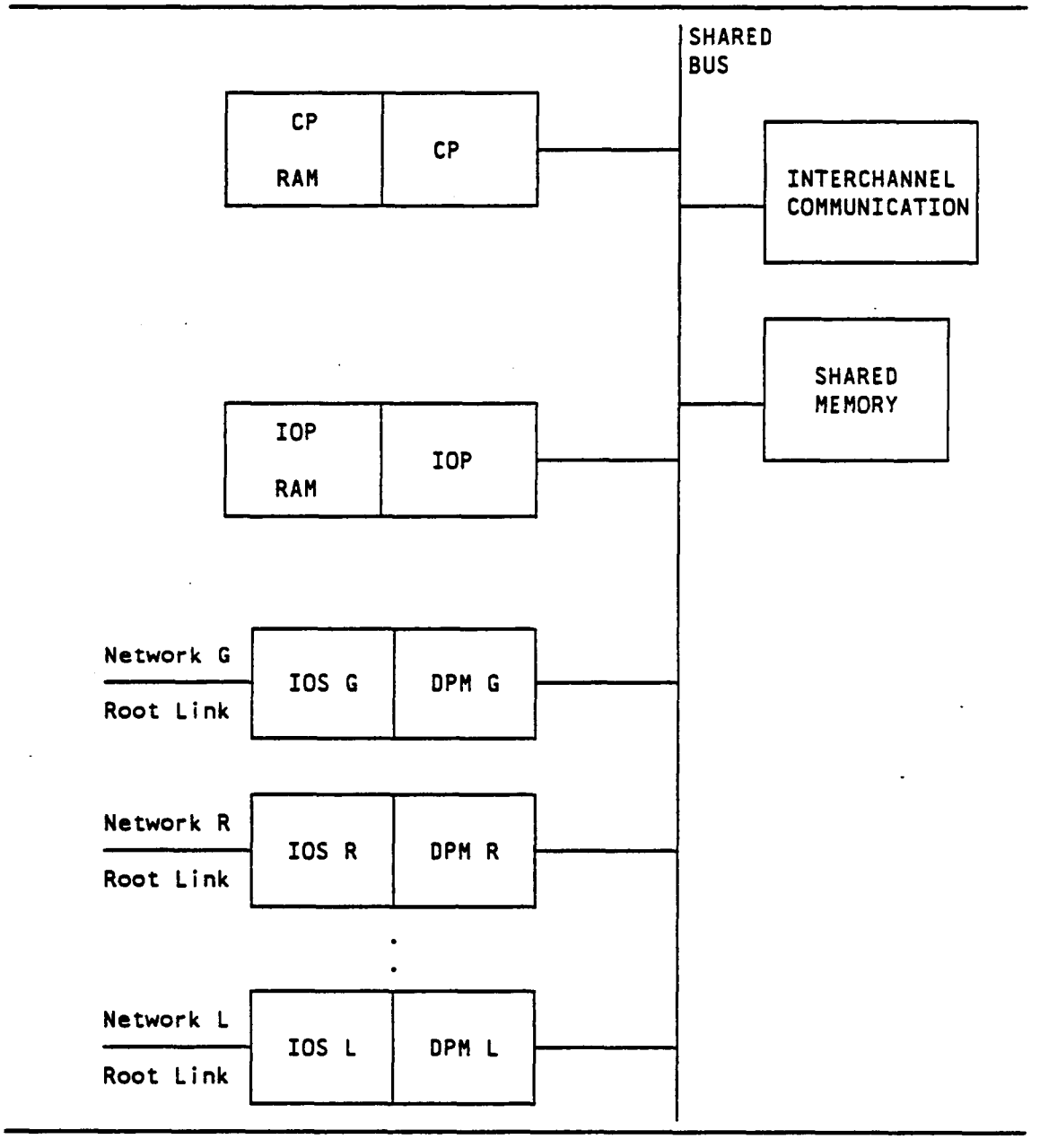


Figure 5-1. FTP - Functional Layout of Major Elements

5.1 Process Location

This section identifies the physical location of the the lowest level processes in each branch of the hierarchy as described in section

" 3. Process Descriptions" on page 3-1. The physical location can be within an FTP: a CP, IOP, or IOS; or external to an FTP: a node.

Those processes that are located in the IOS or node are considered hardware (firmware) processes whereas those located in the CP or IOP are considered software.

Most of the processing is located in either the CP or the IOS. The reasons for the partitioning are:

(1) It is desired to have the IOP capable of a short reaction time, both to chain activation commands from the CP and chain completion commands from the IOS. The intention is to provide a fast I/O response and a small transport lag. Since neither of these two signals is an interrupt to the IOP, the IOP must be monitoring for their occurrence frequently. And obviously, the less other processing the IOP is concerned with, the better its response to these two signals will be.

(2) The IOP will have additional processing assigned related to IC services and FDIR. Since these processes are expected to be more demanding than the I/O processing, it seems appropriate to place modest demands on the IOP for I/O.

(3) It is desired to keep the IOP software independent of the functions assigned to the GPC in order to avoid the necessity of reconfiguring the IOP software during function migration.

This partitioning is an initial set; as experience is gained, it may prove possible to achieve sufficient I/O performance with additional processing allocated to the IOP.

5.1.1 CP Processes

- Request Initiation Processing (4.1.1.1)
- Request Completion Processing (4.1.1.2)
- Sequencer (4.1.2.1.1)
- Chain Initiation (4.1.2.1.2)
- Root Link (4.1.2.1.3)
- Bypass Clear (4.1.2.1.4)
- Transaction Selection (4.1.2.1.5)
- Network Partition (4.1.2.1.6)
- Data/Status Processing (4.1.2.2.1)
- End Of Chain I/O Error Processing (4.1.2.2.2.2)
- GPC I/O Network Manager Support (4.1.4)
- Handle Network Redefinition Events (4.2.1.1.1)
- Send Current Partition Data (4.2.1.1.2)
- Select Root Link (4.2.1.2.1)
- Grow To Root Node (4.2.1.2.2)
- Complete Partition Growth (4.2.1.2.3)
- Repair Network Fault (4.2.1.3.1)
- Test Network Components (4.2.1.3.2)
- Collect Network Status (4.2.2.1)

- Collect Subscriber Status (4.2.2.2)
- Report To GPC Subscribers (4.2.3.1)
- Report To System Manager (4.2.3.2)

5.1.2 IOP Processes

- End Of Chain Monitor (4.1.2.2.2.1)
- Chain Interface (4.1.2.3)
- Interface Chain Setup (4.1.3.1.1)
- Interface Transaction Setup (4.1.3.1.2)

5.1.3 IOS Processes

- Interface Chain Completion (4.1.3.2.1)
- Interface Response Frame Processing (4.1.3.2.2)
- Contention Processing (4.1.3.3)
- Frame Transmitter (4.1.3.4)
- Frame Receiver (4.1.3.5)

5.1.4 Node Processes

- Node (4.1.5)

5.2 Data Location

This section identifies the physical location of each data item listed as the input or output of a process. The location can be one or more of the following FTP elements: IOS dual port memory (DPM), CP memory (CP RAM), IOP memory (IOP RAM), or Shared Memory.

The following physical data locations correspond to the process locations detailed in "5.1 Process Location." Should the processes be relocated, the data locations must be changed accordingly.

5.2.1 CP Memory

The following are data items that exist in congruent form only.

- Active_Root_Node
- Bypass_Clear_Queue_Element
- Chain_Completion_Status
- Chain_Data_And_Status
- Chain_Identifier
- Chain_Queue
- Chain_Transaction_Status
- Current_Network_Definition
- Current_Root_Link

- GPC_Subscriber_Command
- GPC_Subscriber_IO_Error_Log
- GPC_Subscriber_IO_Error_Report
- GPC_Subscriber_List
- IO_Data_Base
- IO_Network_Command
- IO_Network_Manager_Command
- IO_Network_Status_Report
- IO_Request_Data_And_Status
- IO_Request_Definition
- IO_Request_Parameter
- Limits_Definition
- Link_Status
- Local_IO_Status
- Manager_IO_Request_Data_And_Status
- Manager_IO_Request_Parameter
- Network_Configuration
- Network_Definition
- Network_Fault_Indicator
- Network_Identifier
- Network_Initialization_Status
- Network_Partition_Queue_Element
- Network_Status
- Node_Configuration
- Node_Status
- Packet_Status
- Partition_Status
- Predecessor_List.
- Reconfiguration_Report
- Reconfiguration_Status
- Regrow_Network_Request
- Root_Link_Queue_Element
- Root_Link_Status
- Selection_Queue_Element
- Send_Status
- Transaction_Queue_Element
- Transaction_Status
- Wait_Request

5.2.2 IOP Memory

The following are data items that exist in congruent form only.

- Chain_Initiation_Command
- Interface_Transaction_Data

5.2.3 IOS Dual Port Memory

The following are data items that exist in congruent form only. These data items originate in the CP or IOP, where they are congruent. However, the actual usage of one of these items may be by a single channel, and congruency is not relevant.

- Contention_Data
- HDLC_Program
- Output_Packet
- Receiver_State

The following are data items that exist in simplex form only, either in IOS dual port memory or in hardware associated with the IOS.

- Command_Frame
- Contention_Status
- End_Of_Chain_Status
- Input_Packet
- Interface_Transaction_Status
- Response_Frame
- Response_Frame_Data_And_Status

The following are data items that can exist in either simplex or congruent form depending on whether or not the I/O network associated with the related IOS has multiple partitions.

- Command_Frame_Status
- End_Of_Chain_Status_Identifier
- Output_Packet_Identifier

5.2.4 Shared Memory

The following data items must be located in a memory that is accessible by both the CP and the IOP. This can either be a physically distinct shared memory or IOS DPMs. The data is congruent.

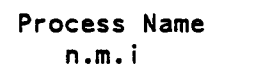
- Activate_Chain
- Chain_Status
- Chain_Timeout_Value
- Completion_Status
- Current_Network_Partition_Definition
- IO_Chain_Definition
- Root_Link_Command
- Transaction_Configuration_Data_Base

APPENDIX A

DATA FLOW DIAGRAM SYMBOL DEFINITIONS

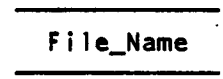
This appendix contains a definition of the symbols used on the data flow diagrams. These symbols are adapted from [10].

This is the process symbol, a circle. It identifies a transformation of input data into output data. The Process Name and Reference Number (see appendix " B. Process Description Format Explanation") appear inside the circle.

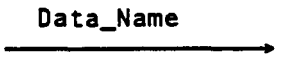


This is the data originator/terminator symbol, a rectangle. It represents a boundary of the system being described. The name of the boundary element appears inside the rectangle.

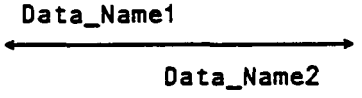
This symbol, two parallel lines, represents a file or data store. The symbol indicates the possibility of a delay when accessing its contents. The name of the store or file appears between the two lines. By convention, both lines are drawn only for the first time the store is identified within a hierarchy; thereafter only one line is drawn.



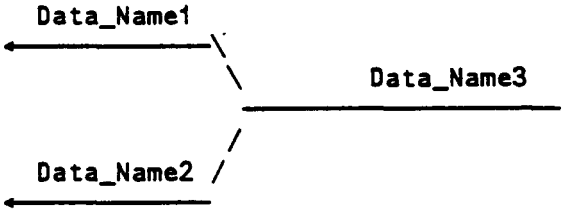
This is the data flow symbol, a directed arrow. The name of the data is written through or next to the line.



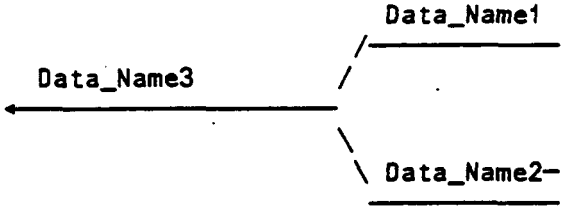
This is the two-way data flow symbol, a double ended arrow. The data flows are separate and should be considered as independent. The two names of the data are written through or next to the line. The proximity of the data name to the arrow end indicates the flow direction of that data.



This is the data flow divergence symbol, branching arrows. This symbol indicates the distribution of data with no processing or transformation taking place. The data may be distributed in total or component data flows can be extracted from the main flow. In the latter case, the names of both the component data and the total data must be indicated.



This is the data flow convergence symbol, merging arrows. This symbol indicates the merging of data with no processing or transformation taking place. The names of both the component data and the total collection of data must be indicated.



APPENDIX B

PROCESS DESCRIPTION FORMAT EXPLANATION

This appendix explains the format of the process descriptions in section " 3. Process Descriptions."

Process Name: This is the name of the process exactly as it appears on the data flow diagram. Each word of the name is capitalized. The words are optionally separated by blanks or underscores.

Reference Number: This is the number of the process exactly at it appears on the data flow diagram.

Identifier: This the Process Name with underscores, fully qualified using the 'dot' notation. This is referred to as the Process ID.

Build: This is the software build in which the process is initially required. For this version of this document entries of either 3 or post 3 are permissible.

Requirements Reference: This entry or entries identifies the paragraph(s) of the appropriate requirements documents [1], [2] , and [3].

Inputs: This list identifies all data inputs to the process by exactly the same names that appear on the data flow diagrams. Every word is capitalized and words are separated by underscores. The source of the input is identified. The source can be either a file or a process.

Outputs: This list identifies all data outputs to the process by exactly the same names that appear on the data flow diagrams. Every word is capitalized and words are separated by underscores. The destination of the output is identified. The destination can be either a file or a process.

Notes: Any special design requirements are entered here. A specific processing rate is an example of a special requirement.

Description:

The description of the processing can be in any combination of several forms: plain english, pseudocode, structured flow diagrams, or a Program Design Language (PDL). In addition, the following conventions are followed:

(1) Figure titles are in the same form as the Process Name.

(2) References to Process Names or IDs use either the fully qualified Process ID, or 'Process Name (Reference Number)'.

(3) References to Data Identifiers use the fully qualified names where necessary to avoid ambiguity; otherwise the simple name is used. For references within flow charts, the simple name is used and text is included in the Description to correlate the simple name to the fully qualified name.

(4) Every input and output item must be referred to in the Description.

APPENDIX C

GLOSSARY OF I/O TERMS

I/O Service Request
Either an request for DIU I/O, a request for Node I/O with contention, or a request for Node I/O without contention.

Node Request Without Contention
A special request for I/O service on exactly one network from the manager of that network consisting of one chained transaction. The individual transactions in the chain involve only nodes; they cannot involve DIUs.

Node Request With Contention
A request for I/O service on exactly one network from the manager of that network consisting of one chained transaction. The individual transactions in the chain involve only nodes; they cannot involve DIUs.

DIU I/O Service Request
A request for I/O service from any user consisting of one or more chained transactions on one or more I/O networks. The individual transactions in the chains involve only DIUs; they cannot involve nodes.

Chained Transaction
A series of one or more transactions on exactly one I/O network.

Transaction
A term that refers to any of the following forms of transactions: input, output, and output/input.

Input Transaction
This type of transaction consists of a command frame followed by a response frame. The predominant information flow is from a node or DIU to a GPC.

Output Transaction
This type of transaction consists of a command frame only. The information flow is from a GPC to a node or DIU.

Output/Input Transaction
This type of transaction consists of a command frame followed by a response frame. There is significant information flow both from the GPC in the command frame and from the DIU or node in the response frame.

Output Transaction with Acknowledgment
This is an output/input transaction in which the data in the response frame contains only the status of the DIU or node.

Frame
A single transmission of data, i.e., a contiguous stream of bits from the opening to the closing flags inclusive.

Command Frame
A frame transmitted by a GPC on an I/O network resulting in the transmission of an output packet.

Response Frame
A frame transmitted by a node or a DIU on an I/O network in response to a received command frame, resulting in the receipt of an input packet by a GPC.

Input Packet
The collected information resulting from a response frame.

Output Packet
The total information transmitted by a command frame.

Contention Sequence
The modified Laning Poll that is performed on a network in order to permit the GPC to perform a chained transaction.

Laning Poll
A contention resolution scheme based on relative priority.

DIU
A Device Interface Unit that interfaces the network to sensors, effectors, and other I/O devices.

Subscriber
A GPC or a DIU connected to a network.

Link
A full duplex, serial transmission path.

Net Link
A link connecting two nodes.

Root Link
A link connecting a GPC to a node.

DIU Link
A link connecting a DIU to a node.

Node

A unit of equipment. that steers transmissions between nodes or between nodes and subscribers

I/O Network

A fault-tolerant, reconfigurable connection between subscribers. The network is made up a number of 5-ported circuit switched nodes. The nodes are interconnected via net links. Subscribers are connected to the network via root links (GPCs) and DIU links.

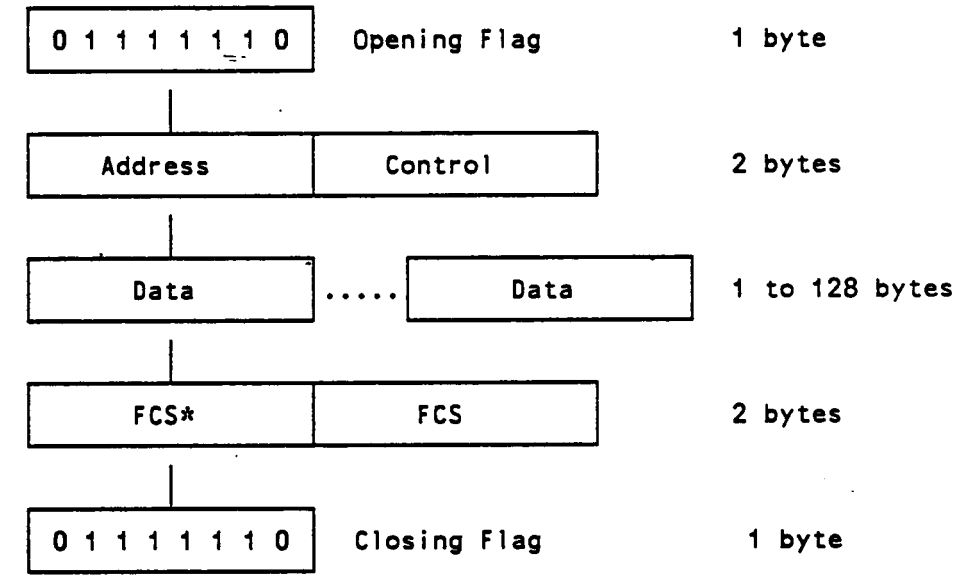
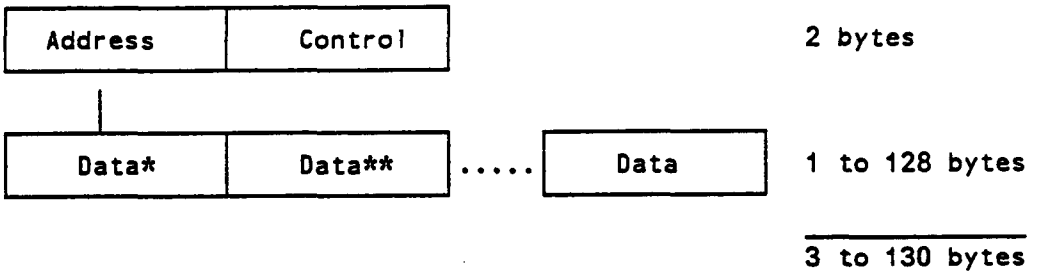


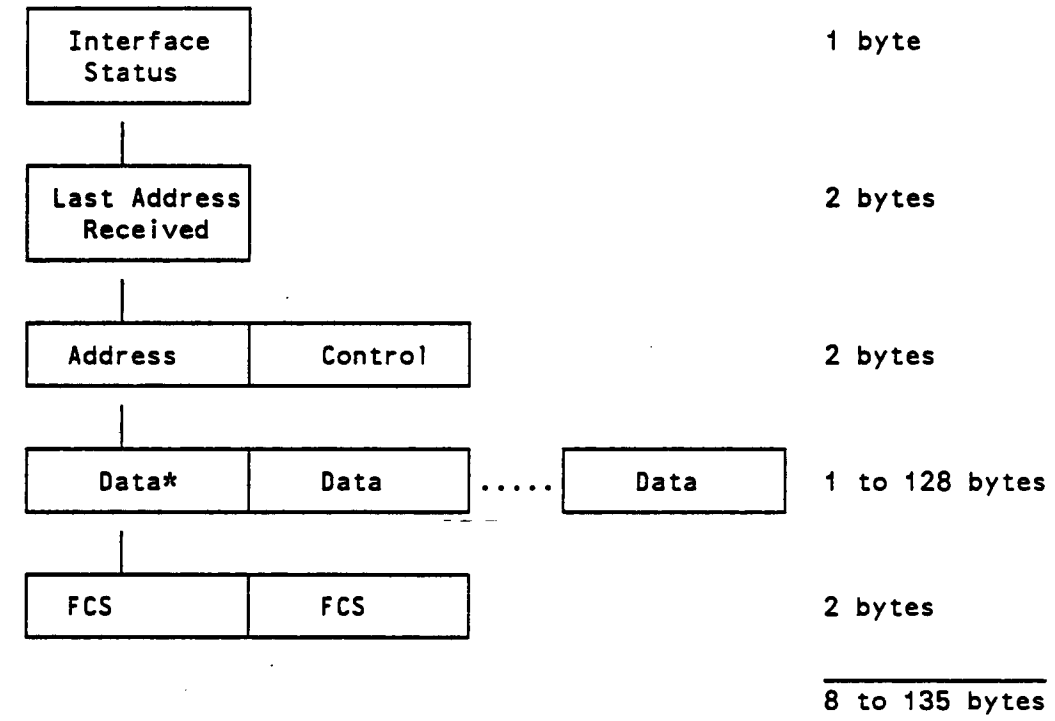
Figure C-1. Frame Format



* For an output packet addressed to a DIU, the one's complement of the DIU address will be contained in the first byte of the data field.

** For an output packet addressed to a DIU, the subcontrol information followed by its one's complement will begin in the second byte of the data field. The exact number of bytes of subcontrol information is DIU specific.

Figure C-2. Output Packet Format



* For an input packet from a DIU, the first byte of the data field will contain the one's complement of the DIU address.

Figure C-3. Input Packet Format

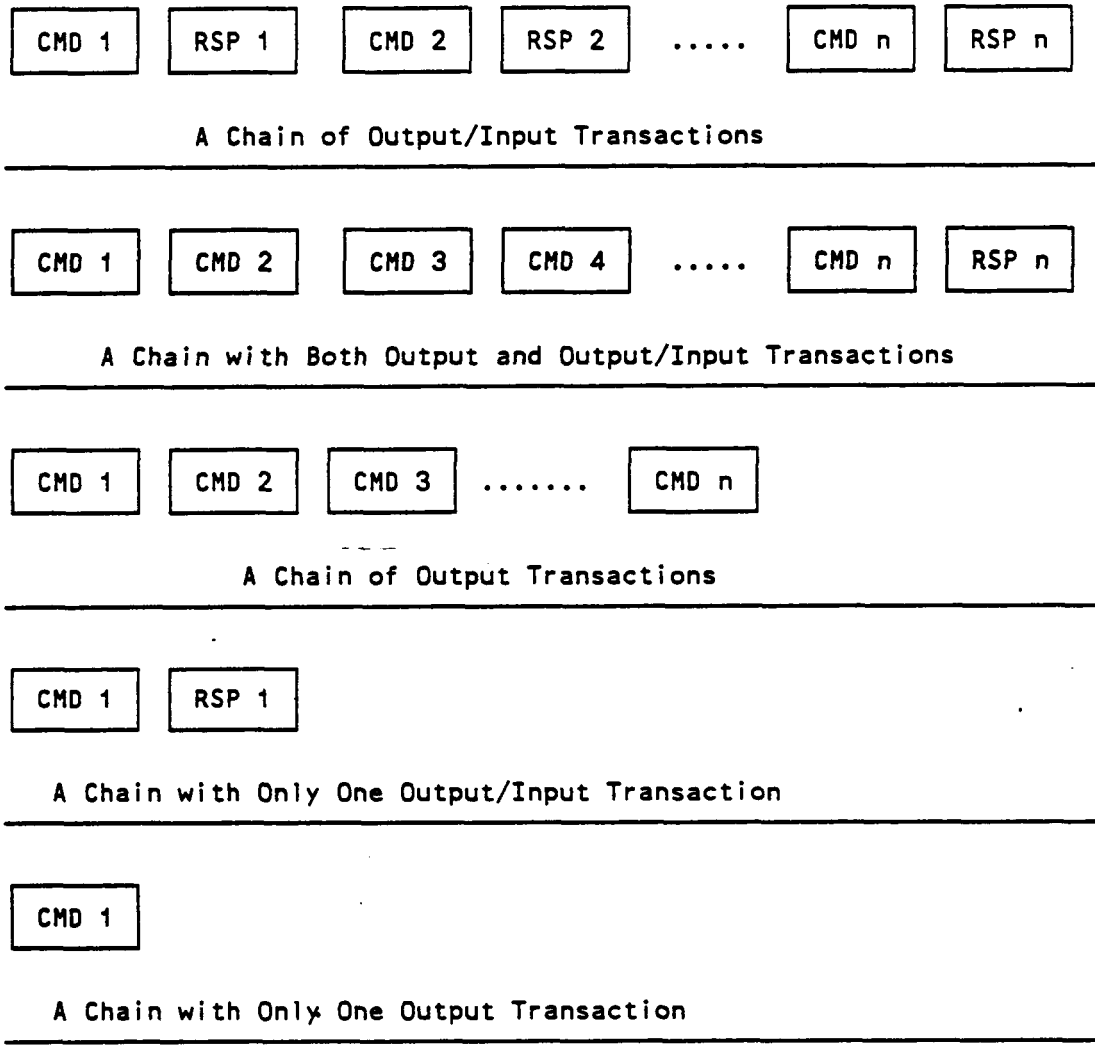


Figure C-4. Chained Transactions

LIST OF REFERENCES

1. Malcolm W. Johnston, James E. Kernan, Jaynarayan H. Lala, and James G. Allen, AIPS System Requirements (Revision 1)., CSDL Report No. CSDL-C-5738, August 30, 1983.

2. Advanced Information Processing System (AIPS) System Specification, CSDL Report No. CSDL-C-5709, (Revision 1), October 1984.

3. Proof of Concept I/O Network System Services Functional Requirements, CSDL-AIPS-84-138, December 6, 1984.

4. J. McKenna Jr, et al, Advanced Information Processing System (AIPS) Proof of Concept System Design Specification, Fault Tolerant Processor, CSDL-AIPS-84-161.

5. J. McKenna Jr, et al, Advanced Information Processing System (AIPS) Proof of Concept System Design Specification, Network Node, CSDL-AIPS-84-162.

6. Eliezer Gai, et al, AIPS Methodology Report, CSDL-C-5699, May 1984, pages 2-1 through 2-8.

7. Larry D. Brock J. Barton DeWolf, Proposed Requirements Methodology For Integrated Avionics Systems, CSDL-R-1656, July 28, 1983.

8. Asok Ray, Richard Harper, Jaynarayan H. Lala, Damage-Tolerant Multiplex Bus Concepts For Far-Term Digital Flight Control Systems, CSDL-R-1690, February, 1984.

9. Norman L. Kerth, "Software Tools Automate Structured Analysis", Electronics Week, August 20, 1984, Pages 69-72.

10. Brian Dickinson, Developing Structured Systems, Yourdon Press, 1980.

M

N

S

T

W