ANSI®
TR X3.xxx-199x
Revision 9 of
X3-991D

draft proposed X3 Technical Report —

Small Computer System Interface - 3 Generic Packetized Protocol (SCSI-GPP)

Secretariat

Computer and Business Equipment Manufacturers Association

Approved xxxx xx, 199x

X3 Technical Committee of American National Standards Institute

Abstract

SCSI-3 Generic Packetized Protocol (SCSI-3 GPP) defines a transport protocol that permits an inter-operable common mapping of the Small Computer System Interface - 3 (SCSI-3) functions to several physical interfaces that do and do not have a native SCSI-3 protocol mapping. In support of such heterogeneous systems environments (e.g., SCSI-3 parallel interface, Fibre Channel, and Internet), some implementors may choose to use GPP as an alternate protocol to provide common software and transportation services across several interfaces.

X3's Technical Report Series

This Technical Report is one in a series produced by the American National Standards Committee, X3, Information Technology. The secretariat for X3 is held by the Computer and Business Equipment Manufacturers Association (CBEMA), 1250 Eye Street, NW Suite 200, Washington DC 20005.

As a by-product of the standards development process and the resources of knowledge devoted to it, X3 from time to time produces Technical Reports. Such Technical Reports are not standards, nor are they intended to be used as such.

X3 Technical Reports are produced in some cases to disseminate the technical and logical concepts reflected in standards already published or under development. In other cases, they derive from studies in areas where it is found premature to develop a standard due to a still changing technology, or inappropriate to develop a rigorous standard due to the existence of a number of viable options, the choice of which depends on the user's particular requirements. These Technical Reports, thus, provide guidelines, the use of which can result in greater consistency and coherence of information processing systems.

When the draft Technical Report is completed, the Technical Committee approval process is the same as for a draft standard. Processing by X3 is also similar to that for a draft standard.

PATENT STATEMENT

CAUTION: The developers of this technical report have requested that holder's of patents that may be required for the implementation of the Technical Report, disclose such patents to the publisher. However, neither the developers nor the publisher have undertaken a patent search in order to identify which, if any, patents may apply to this Technical Report.

As of the date of publication of this Technical Report and following calls for the identification of patents that may be required for the implementation of the Technical Report, no such claims have been made. No further patent search is conducted by the developer or the publisher in respect to any Technical Report it processes. No representation is made or implied that licenses are not required to avoid infringement in the use of this Technical Report.

Published by

American National Standards Institute
1430 Broadway, New York, New York 10018

Printed in the United States of America

Contents

Page

Tables

Page

Figures

Page

Foreword (This Foreword is not part of X3 Technical Report X3.xxx-199x.)

SCSI-3 Generic Packetized Protocol (SCSI-3 GPP) is a SCSI protocol that permits an inter-operable common mapping of the Small Computer System Interface (SCSI) functions to several physical interfaces that do and do not have a native SCSI-3 mapping. In support of such a heterogeneous system environment (e.g., SCSI-3 parallel interface, Fibre Channel, and Internet), some implementors may choose to use GPP as single common transport protocol.

This technical report was developed by Technical Committee X3T10 of Accredited Standards Committee X3 starting in 1990. The technical report approval process started in 1994. this document included annexes which are informative and are not considered part of the technical report.

Requests for interpretation, suggestions for improvement or addenda, or defect reports are welcome. They should be sent to the X3 Secretariat, Computer and Business Equipment Manufacturers Association, 1250 Eye Street, NW, Suite 200, Washington, DC 20005-3922.

This technical report was processed and approved for submittal to ANSI by Accredited Standards Committee on Information Technology, X3. Committee approval of the technical report does not necessarily imply that all committee members voted for approval. At the time it approved this technical report, the X3 Committee had the following members:

xxx, Chair
xxx, Vice-Chair
xxx, Secretary

Organization Represented — Name of Representative

xxx xxxx

Subcommittee X3T10 on Low Level Interfaces, which reviewed this technical report, had the following members:

xxx, Chair
xxx, Vice-Chair
xxx, Secretary

Organization Represented — Name of Representative

xxx xxxx

The designated editor of this document was:

Gary Stephens

Introduction

The Generic Packetized Protocol provides a general information transfer protocol between SCSI devices using a structure called an Information Packet. The command descriptor blocks, logical data, status, and autosense information are described in the SCSI-3 Architecture Model (SAM) standard and one or more SCSI-3 command set standards. The various data types are placed in a logical order into Information Packets, relative to a Task, and transmitted between a Logical Unit and an Initiator. GPP defines the service delivery mapping to a physical interface.

The Information Packets of the Generic Packetized Protocol are capable of being transported over any physical medium which provides delivery of an ordered sequences of bytes. GPP is appropriate for heterogeneous interconnect environments where several different transport systems may exist in a system or where two or more different transport systems must be traversed between an Initiator and a Target (e.g., in the Heterogeneous InterConnect standard and in Internet). Additionally, GPP may be used for efficient communication in environments where there may be a long distance between an Initiator and a Target (e.g., Fibre Channel, Internet, or Asynchronous Transfer Mode (ATM)).

GPP specifies the appropriate transport layer protocols for those interfaces where heterogeneous interconnects may be used to form a single service delivery subsystem. In particular, GPP specifies mappings for performing SCSI-3 Tasks:

Over Fibre Channel (FC-PH) using any of its topologies,
Over the SCSI-3 Parallel Interface (SPI),
Over the various physical layers specified by the Heterogeneous InterConnect standard (IEEE 1355),
Over any transport layer capable of supporting the Internet Protocol (IP),
Over Asynchronous Transfer Mode (ATM/AAL5),
and combinations of these individual transport layers.

Mapping to other service delivery subsystems is outside the scope of this X3 Technical Report.

X3 Technical Report for Information Technology

SCSI-3 Generic Packetized Protocol (SCSI-3 GPP)

1 Scope

The SCSI-3 Generic Packetized Protocol (GPP) permits a heterogeneous interconnection of multiple Initiators and multiple Targets. GPP provides a common mapping that is used across all supported physical interfaces. This permits a single software interface in both Initiators and Targets independent of the physical interface. GPP also provides control mechanisms to use multiple paths between two devices during a single Task. GPP permits the construction of high availability systems where the loss of a port or link in the transport layer need not cause Tasks to be aborted but rather they may be continued on some other part of the service delivery subsystem.

1.1 Application

GPP is appropriate

- where a common mapping layer is desirable in a system with multiple physical interfaces supporting SCSI-3 (i.e., all initiators and targets have a single interpretation requirement for data transfer independent of the physical interface)

- for heterogeneous interconnect systems, where there may be several different physical layers between the application client and the logical unit

- where there is a long distance between an Initiator and a Target or where there the overall throughput per task is important.

Examples of these different uses for GPP are given below:

- GPP may be used in a Fibre Channel environment where there is a complex fabric or an arbitrated loop between two nodes. The GPP protocol reduces the effect of latencies introduced in such an environment.

- GPP may be used in an environment where the service delivery subsystem between the Initiator and Target consists of several different transport layers, such as two nodes attached to the Internet separated by Ethernet and Token Ring segments, plus bridges and routers.

- GPP may be used in a large, complex Asynchronous Transfer Mode (ATM) network where two nodes of the network are separated by considerable distance (e.g., tens or hundreds of kilometers). In this case, the efficiency of the GPP protocol permits fast transfer of SCSI information because the protocol reduces the number of turnarounds on a virtual circuit.

GPP permits a system integrator to form larger, more diverse system structures and still use SCSI-3. Using GPP, SCSI-3 devices that have specific SCSI-3 protocols defined by SCSI-3 protocol standards may be accessed through a bridge across these heterogeneous interconnects. GPP permits a single service delivery subsystem, as specified in SAM and composed of different segments selected from the supported physical interfaces, to gain access to the stored information using the native SCSI-3 commands sets.

1.2 Structure

The overall structure of GPP is outlined in the below.

Clause 2 provides a reference list of reference standards and other documents.

Clause 3 provides a comprehensive glossary of terms used in GPP.

Clause 4 provides a model for operations and mandatory requirements for logical operation of GPP.

Clause 5 provides the mandatory requirements to manage a task with GPP.

Clause 6 defines the method for encapsulating logical functions in Information Packets.

Clause 7 defines the primary messages and protocols for their use to support the Information Packet structure on all physical interfaces. Additional messages may be provided for each service delivery subsystem in the appropriate annexes.

Clause 8 specifies a service interface from an application client to GPP.

The remainder of GPP consists of two annexes per physical interface. The first annex of the pair specifies the requirements for encapsulating an Information Packet for transfer on that physical interface. The second annex of the pair gives usage guidelines for that physical interface.

Annex A specifies the mapping of GPP for use with Fibre Channel.

Annex B gives usage guidelines for using GPP with Fibre Channel.

Annex C specifies the mapping of GPP for use with the SCSI-3 Parallel Interface.

Annex D gives usage guidelines for using GPP with the SCSI-3 Parallel Interface.

Annex E specifies the mapping of GPP for use with the Internet protocol.

Annex F gives usage guidelines for using GPP with Internet.

Annex G specifies the mapping of GPP for use with the Heterogeneous InterConnect (HIC) standard (IEEE P1355).

Annex H gives usage guidelines for using GPP on HIC.

Annex I specifies the mapping of GPP for use with the Asynchronous Transfer Mode (ATM) AAL5 layer.

Annex J gives usage guidelines for using GPP with ATM AAL5.

1.3 Relationship to standards

Figure 1 shows the general applicability of SCSI-3 standards, technical reports and related standards to one another, and not a hierarchy, a protocol stack, or a system architecture. For example:

- SCA and SAM and the SCSI-3 command set standards are applicable to all protocols.
- SIP, SSP, FCP, and SBP are link specific native protocols designed to be applied only to the service delivery subsystem directly below each of them.

- GPP (an X3 technical report) is intended to be used with heterogeneous interconnects between devices using the native protocols above and some additional physical interfaces.

The SCSI-3 SCA, SAM, and command set standards provide computers with device type independence within a class of devices. GPP provides service delivery subsystem independence for both Initiators and Targets.

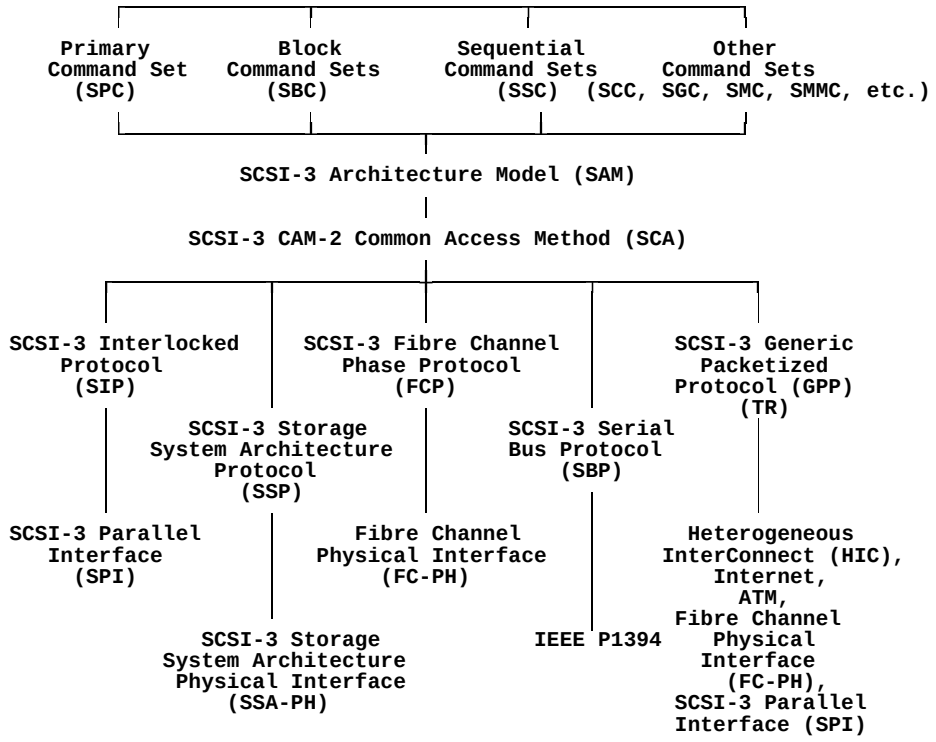


Figure 1 - SCSI-3 road map

The term SCSI is used whenever it is not necessary to distinguish between the versions of SCSI. The original Small Computer System Interface Standard, X3.131-1986, is referred to in GPP as SCSI-1. SCSI-1 was revised and replaced with the Small Computer System Interface - 2 , X3.131-1994, which is referred to in GPP as SCSI-2.

The term SCSI-3 refers collectively to the following documents:

Architecture

SCSI-3 Architecture Model (SAM), X3-994D

Protocols

SCSI-3 Interlocked Protocol (SIP), X3-856D

SCSI-3 Fiber Channel Protocol (FCP), X3-993D

SCSI-3 Serial Bus Protocol (SBP), X3-992D

SCSI-3 Serial Storage Architecture Protocol (SSP), X3-1051D

SCSI-3 Generic Packetized Protocol (GPP), X3-991D (ANSI X3 Technical Report)

Command Sets

SCSI-3 Primary Commands (SPC), X3-995D

SCSI-3 Block Commands (SBC), X3-996D

SCSI-3 Stream Commands (SSC), X3-997D

SCSI-3 Graphic Commands (SGC), X3-998D

SCSI-3 Medium Changer Commands (SMC), X3-999D

SCSI-3 Controller Commands (SCC), X3-1047D

SCSI-3 Multi-Media Commands (SMMC), X3-1048D

Host Software Interfaces

SCSI-2 Common Access Method (CAM), X3.232-199x/792D

Physical Interfaces

SCSI-3 Parallel Interface (SPI), X3.253-199x/855D.

2 References

The following standards contain provisions which, through reference in GPP, are provisions of GPP. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the following list of standards. Members of IEC and ISO maintain registers of currently valid International Standards. ANSI performs a similar function for American National Standards.

ANSI X3.4-1977, *American Standard Code for Information Interchange (ASCII)*.

ANSI X3.xxx-199x, *SCSI-3 Architecture Model (SAM)*.

The following standards are listed to help implementers evaluate the complete operating environment for GPP devices.

SCSI-3 related standards

[1] ANSI X3.xxx-199x, *SCSI-3 Parallel Interface (SPI)*.

[2] ANSI X3.xxx-199x, *SCSI-3 Interlocked Protocol (SIP)*.

[3] ANSI X3.xxx-199x, *SCSI-3 Primary Command Sets (SPC)*.

[4] ANSI X3.xxx-199x, *SCSI-3 Common Access Method-2 (SCA)*.

Fibre Channel related standards

[5] ANSI X3.230-199x, *Fibre Channel - Physical and Signaling Interface (FC-PH)*.

[6] ANSI X3.xxx-199x, *Fibre Channel - Fabric Requirements (FC-FG)*.

[7] ANSI X3.xxx-199x, *Fibre Channel - Switching and Switch Controls Topology (FC-SW)*.

[8] ANSI X3.xxx-199x, *Fibre Channel - Arbitrated Loop Topology (FC-AL)*.

Other interface standards

[9] IEEE 1355, *Heterogeneous InterConnect (HIC)*.

Asynchronous Transfer Mode (ATM)

[11] ITU-T Recommendation I.361-1993, *BISDN ATM Layer Specification.*

[12] ITU-T Recommendation I.363-1993, *BISDN ATM Adaptation Layer (AAL) Specification.*

[13] ATM Forum, *User-Network Interface Specification (UNI)*.

Internet

[14] RFC 791, *Internet Protocol*.

[15] RFC 792, *Internet Control Message Protocol*.

[16] RFC 793, *Transmission Control Protocol*.

[17] RFC 1060, *Assigned Numbers*.

[18] RFC 1191, *Path MTU Discovery*.

3 Definitions, abbreviations and symbols

3.1 Definitions

This clause contains a glossary of special terms used in this Technical Report. These terms apply to GPP and terms do not constitute a comprehensive glossary for SCSI-3. Additional glossaries may be found in in the annexes for each service delivery interface to define terms used only in those annexes.

3.1.1 bit: A single binary digit (bit).

3.1.2 burst: A byte string representing a portion of a large data structure and placed in an interface logical element.

3.1.3 byte: An 8-bit construct.

3.1.4 byte string: A contiguous ordered set of bytes.

3.1.5 connect: The process which transfers an Information Packet between an Initiator and a Target. In GPP, the role of the port changes depending on the physical direction of the Information Packet flow. A connect requires a successful Information Packet transfer associated with a Task.

3.1.6 field: A set of one or more contiguous bits.

3.1.7 GPP address: The unique SCSI device address for one port on an GPP device relative to one service delivery interface. The GPP address is unique on a service delivery interface. Multiple ports on the same GPP device connected to the same service delivery interface have different GPP addresses. The form of the GPP address varies with the service delivery interface used.

3.1.8 GPP device: A SCSI device that uses the Generic Packetized Protocol with a service delivery subsystem. A device capable of attaching to one or more GPP interfaces via one or more ports, respectively. A GPP device may allow a port to function in either Initiator or Target for any one connect. Each GPP port may operate in a different role for each different connect.

3.1.9 GPP interface: The set of all GPP devices to which a single GPP port can communicate and the interconnecting transmission media.

3.1.10 GPP port: Port.

3.1.11 H_C relationship: A relationship between an Initiator and a Target that begins with an initial connect and ends with the completion of activity associated with the Task or Task Management function.

3.1.12 H_C_L relationship: A relationship between an Initiator, a Target, and a Logical Unit that begins with an initial connect and ends with the completion of activity associated with the Task or Task Management function.

3.1.13 H_C_L_Q relationship: A relationship between an Initiator, a Target, a Logical Unit, and a tag that begins with an initial connect and ends with the completion of activity associated with the Task or Task Management function.

3.1.14 H_C_x_y relationship: A relationship which is either an H_C_L or H_C_L_Q relationship that begins with an initial connect and ends with the completion of activity associated with the Task or Task Management function.

3.1.15 H_I_T_C relationship: An H_C relationship, or one of its derivatives, between an Initiator and a Target that operates in single path mode that begins with an initial connect and ends with the completion of activity associated with the Task or Task Management function.

3.1.16 identify function: A process performed by logical elements to identify a task. The result of the process is a correctly formed set of interface control prefix fields.

3.1.17 Information Packet: A set of bytes containing Interface Control Fields and at least one Interface Logical Element.

3.1.18 initial connection: A connect in which an Initiator establishes a relationship in a Target.

3.1.19 Initiator: A logical element which normally starts Tasks. Tasks execute using the services of one or more ports in the Initiator.

3.1.20 Initiator role: The operating mode of a port which permits it to send Information Packets.

3.1.21 Initiator mode port: A port operating in Initiator role. An Initiator mode port attaches to exactly one service delivery interface.

3.1.22 Interface Control Fields: An organized collection of bytes used to encapsulate Interface Logical Elements into Information Packets. GPP interprets some of these fields to accomplish proper routing and sequencing of Information Packets between Initiators and Targets.

3.1.23 Interface Logical Element: An self-identifying structure used to communicate the logical content of SCSI using the Generic Packetized Protocol.

3.1.24 invalid: An illegal, reserved, or unsupported bit, field, code value, or protocol sequence.

3.1.25 logical element: An Initiator or a Target.

3.1.26 logical system: From the point of view of a single logical element, the set of unique GPP devices reachable through all ports of a logical element. For any two logical elements which can reach each other on at least one GPP interface, the set of other reachable GPP devices each identifies may not be the same. The same logical element may be reached through more than one port.

3.1.27 Logical Unit: An addressable function in a Target that implements a device model.

3.1.28 one: A true signal value, a field with a value of 1b, or a field value numerically equal to 1b.

3.1.29 path: A path is a named interconnect between an Initiator and a Target. At least one connect must be made from the Initiator to the Target before an interconnect becomes a path.

3.1.30 port: A port is a portion of the service delivery interface of a GPP device. A GPP device may have more than one port. Each port may attach to the same or a different GPP interface. Each port has a port number assigned by to its logical element and also one or more GPP addresses.

3.1.31 port number: A unique number for each port in a logical element assigned by the controlling logical element.

3.1.32 receive (an Information Packet): The result of the successful transfer of an Information Packet between two logical elements.

3.1.33 reconnect: The act of reviving a relationship to continue a Task. The connection may be on the same path as the initial connection or on a different path when using Multiple Path Mode. A reconnect completes when conditions are appropriate to begin transfer of an Information Packet.

3.1.34 reserved: The term used for fields, code values, and other items set aside for future definition.

3.1.35 tag: The parameter associated with a Task that identifies a Task when an Initiator starts multiple Tasks for a logical unit. A queue tag is not related to the path or paths used for a Task; it is related only to the Initiator and logical unit.

3.1.36 Target: An attachment point for Logical Units to a service delivery interface.

3.1.37 Target role: The operating mode of a port which permits the port to receive an Information Packet from another port.

3.1.38 Target mode port: A GPP port operating in Target role. An Target attaches to exactly one service delivery interface.

3.1.39 Task: A Task consists of one initial connection and zero or more reconnects with a minimum of one Information Packet transfer per connection. The connect(s) pertain to a relationship in which at least one Information Packet is successfully transferred.

3.1.40 unexpected disconnect: A disconnection which occurs because of a protocol error.

3.1.41 vendor-specific: Something (e.g., a field, code value, etc.) this Technical Report identifies, but its use is not defined by this Technical Report and may be used differently in various implementations.

3.1.42 world-wide name: A unique numeric value assigned to logical elements. The value is unique within a logical system.

3.1.43 WWNi.: A WWNi is an world-wide name assigned to an Initiator. The identifier is communicated to Targets over the interface as part of the multiple path mode option.

3.1.44 WWNt: An WWNt is an world-wide name assigned to an Target. The identifier is communicated to Initiators over the interface as part of the multiple path mode option.

3.1.45 zero: A field with a value of 0b in each bit position.

3.2 Abbreviations and symbols

The following abbreviations are used within GPP.

AAL ATM Adaptation Layer

ACA Auto Contingent Allegiance

AEN Asynchronous event notification

ATM Asynchronous Transfer Mode

CAM Common Access Method

CCB CAM Control Block

CDB command descriptor block

CPD command parameter data

CRD command response data

e.g. for example (illustrative)

FC-PH Fibre Channel Physical and Signaling

HIC Heterogeneous Interconnect (IEEE P1355)

i.e. that is (or said otherwise)

I/O input/output

ID identifier

ILE Interface Logical Element

IP Internet Protocol

ISO International Standards Organization

IU Information Unit

LD logical data

LSB least significant bit

LU Logical Unit

LUN logical unit number

MSB most significant bit

OSD operating system dependent

Pkt Information Packet

SCSI Any version of the Small Computer System Interface standard

SCSI-3 Only standards identified in the foreword as being part of the SCSI-3 standard document set

SDS service delivery subsystem

SPI SCSI-3 Parallel Interface

TCP Transmission Control Protocol

TRB task request block

ULP upper level protocol (FC-PH)

WWN world-wide name

WWNi Initiator world-wide name

WWNt Target world-wide name

The following abbreviations are used within GPP.

|| Concatenation; merging two items to form one new item

.. a domain of integer values (a..b) bounded by a and b, a < b

4 General

This clause specifies the logical system for a GPP environment where device-independent activities using one or more GPP service delivery subsystems may be performed. The activities performed are called logical operations.

A service delivery subsystem is not an integral part of this portion of the GPP environment. GPP specifies a service delivery subsystem independent mechanism for carrying out logical operations. A service delivery subsystem is required, but its function is separately defined from GPP logical operations.

4.1 Overview

Logical operations consists of Tasks and Task management that are accomplished by transferring Information Packets composed of interface logical elements (ILEs). ILEs are messages, command descriptor blocks, command parameter data (additional data required for some commands), command response data (data not from logical data), logical data, autosense, and status.

Initiators and Logical Units send, receive, and process Information Packets. They interpret the ILEs contained in each Information Packet, prepare appropriate responses in Information Packets, if any, and send the response Information Packets.

4.2 Conventions

Certain words and terms used in GPP have specific meanings beyond the normal English meaning. Glossaries define these words and terms (see 3.1 and the annexes) or the definition appears at first use in the text. Names of signals, phases, and messages are in all upper-case letters (e.g., INVALID Information Packet).

Words have the normal technical English definition unless the word or phrase is defined in context or in one of the glossaries, then that definition is used. Some words may have definitions unique to the clauses where they are used. Every effort has been made to limit the number of such instances.

Numbered items in GPP do not represent any priority. Any priority is explicitly indicated.

In all of the text, tables, and figures, the most significant bit of a field is shown on the left side and represents the highest algebraic value position in the quantity.

The word "shall" is used to indicate a mandatory requirement. If such a requirement is not followed, the results are unpredictable unless indicated otherwise. In addition, if such a requirement is not followed, the implementation is not in conformance with the requirements of GPP.

If a field is specified as not meaningful or is to be ignored, the entity that receives the field shall not check that field.

If a conflict arises between text, tables, and figures, the order of precedence to resolve conflicts is text, tables, and, lastly, figures. Not all tables and figures are fully described in text. Tables show data formats and values; figures are illustrative of the text and tables. NOTEs and IMPLEMENTATION NOTEs have been kept to a minimum, but when they occur they do not constitute any requirements for implementors.

Numbers that are not immediately followed by lower-case "b" or "h" are decimal values.

The ISO convention of numbering is used (i.e., the thousands and higher multiples are separated by a space and a comma is used as the decimal point).

A sequence of one or more of the digits 0 and 1 immediately followed by lower-case "b" are binary values.

A sequence of one or more of the digits 0-9 and the upper-case letters "A"-"F" immediately followed by lower-case "h" are hexadecimal values.

4.3 Logical system description

The SCSI-3 Packetized Protocol, as viewed by Initiators and Logical Units, is shown in figure 2. The Initiator prepares an Information Packet and sends the Information Packet to a Logical Unit using the services of a service delivery subsystem and a Target. The Logical Unit receives the complete Information Packet, interprets it, prepares a response in the form of an Information Packet, and sends the Information Packet through the Target to the Initiator. The Initiator receives a complete Information Packet.

The exchange of Information Packets continues until a Task is complete. Information Packets for multiple Tasks may be multiplexed by both the Initiator and the Logical Unit to the same or different GPP devices.

The direction of flow of Information Packets and their content is determined by the logical operations requested by the Task. For example, some Tasks may request read-type operations from a disk, while other Tasks may print a data base report on a printer.

Since Information Packets are used for all communication, the two transport layers in figure 2 may have different characteristics. For example, one may be Fibre Channel links with an arbitrated loop or a fabric and the other may be a SPI 2-byte wide, fast parallel data bus.

All operations are placed into or extracted from complete Information Packets. All logical interpretation is based on only using complete and correctly transferred Information Packets. This is a buffer-to-buffer transfer model of communications. No information from the service delivery subsystem is required to prepare or interpret Information Packets.

4.4 Description of logical operations

This clause identifies the major elements of GPP Task flow. A Task consists of a minimum of one Information Packet. This type of Task usually requests some Task management function. Such a Task is concerned only with messages, such as the TARGET RESET message, that do not require a response from the Logical Unit.

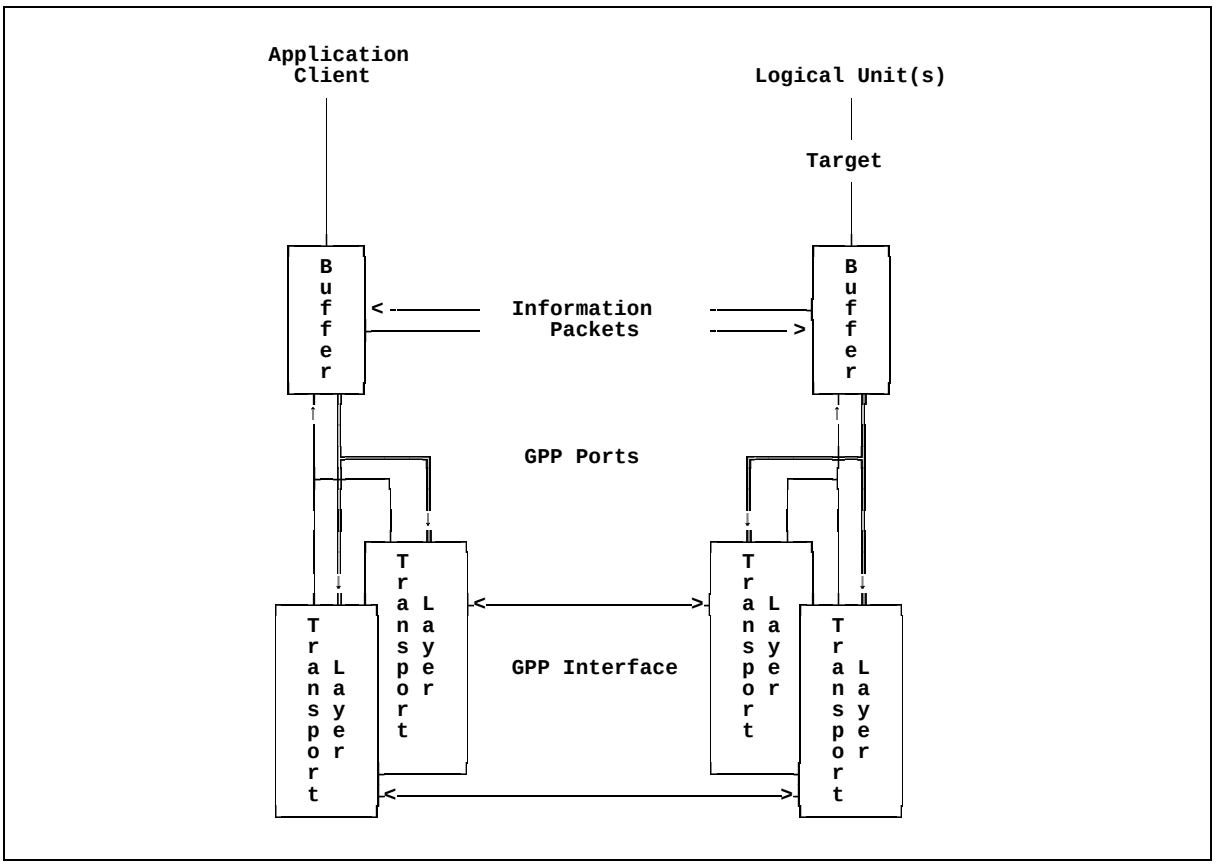


Figure 2 - GPP high level structure

Most Tasks consists of at least two Information Packets. One Information Packet is sent from an Initiator to a Logical Unit to request some function(s) be performed. The Logical Unit responds at completion of the function with an Information Packet indicating the status of the request(s). Tasks normally require an exchange of Information Packets in both directions.

There are different kinds of data transferred in Information Packets. Each different kind of data is called an ILE. The ILEs are command descriptor blocks, status, messages, command parameter data, command response data, autosense, and logical data (see 6.3). A Logical Unit may have multiple ILEs from one or more Information Packets available for a Task from an Initiator before starting a Task or during execution of a Task.

Figure 3 shows the five types of commands defined in SAM and the corresponding ILEs used with each command type. The descriptions below reference the command type by the number in the left column.

The basic request to perform a function is a data structure called a command descriptor block (CDB). Each CDB is fixed length. Some commands need to transfer more data than can fit into the fixed length CDB. This additional data is called command parameter data (see figure 3, type 2). Command parameter data is variable in length, with the length specified in or implied by the CDB.

For commands which do not transfer logical data, the Logical Unit checks the CDB and the command parameter data, if any. If all parameters contain valid values for the state of the Logical Unit, the Logical Unit may execute the command according to the Task management model in SAM. When execution is complete and successful, the Logical Unit forms an Information Packet containing the status of the command and one command completion message (there are three to choose from). The response Information Packet, when received at the Initiator is checked and the Task parameters are updated.

A different type of command has no command parameter data and no logical data. This type of command requires that the Logical Unit return data, other than logical data, as command response data (see figure 3, type 3). Execution is similar to the command description above.

For commands that transfer command parameter data, command response data or logical data, more than two Information Packets may be needed (see figure 3, types 2, 3, 4 and 5).

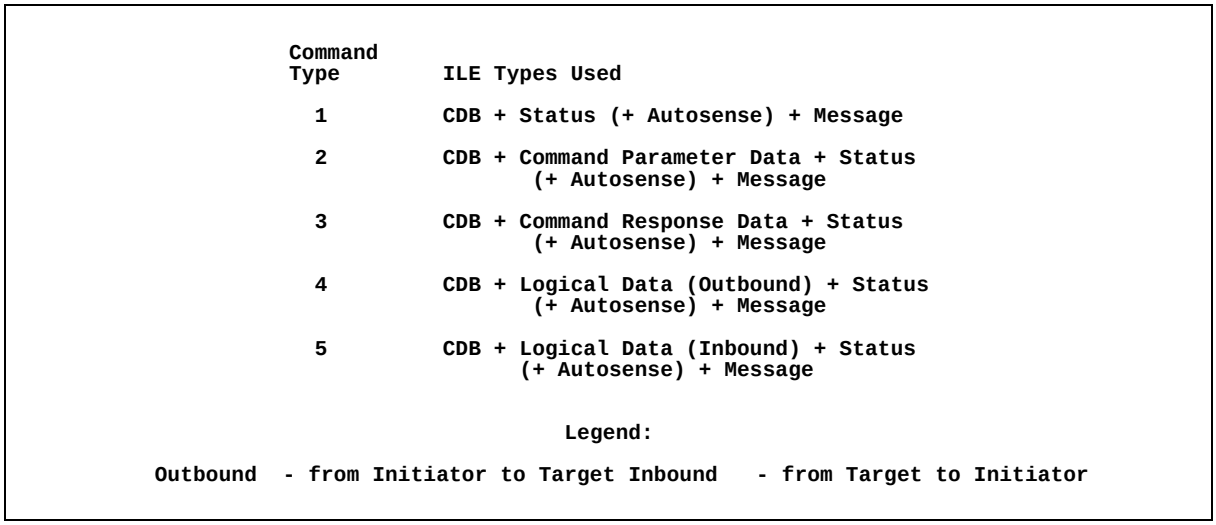


Figure 3 - Command types

For write-type commands, the Initiator places the CDB and possibly some of the logical data in the first Information Packet and sends it. The Logical Unit checks the CDB as before. If all parameters are correct for the current state of the Logical Unit, the Logical Unit may begin to process the logical data, if any is present in the Information Packet.

While Logical Unit processing begins on the first Information Packet, the Initiator may have formed additional Information Packets and sent them, up to the limit agreed upon between the two logical elements. The Logical Unit overlaps receipt of additional Information Packets with processing of each successfully received Information Packet. The Logical Unit informs the Initiator as the Information Packets are processed so that additional packets, if any, can be transferred. The Initiator is informed of progress through status and messages transferred from the Target.

When all logical data has been transferred, the Logical Unit finishes processing the logical data. The status and command completion message are placed in an Information Packet and sent to the Initiator. The command, and in this example the Task, is complete.

The Initiator can link several commands together to form an extended Task for a single Logical Unit and place the commands in one or more Information Packets. Tasks with linked commands are treated as a single unit by the Logical Unit and the Initiator.

If, as is occasionally the case, the CDB block contains an error, the command is not executed. Any ILEs following the CDB block are not processed (i.e., treated as not having been received). Additional Information Packets for the Task are not processed and are discarded. Detecting the error requires the Logical Unit to terminate the Task and prepare sense data to identify the error. The process of reporting and clearing the error is called Auto Contingent Allegiance (ACA) in SAM. GPP supports ACA as specified in SAM.

4.5 Statement of support for SAM requirements

This clause provides a cross reference to the protocol specific requirements stated in SAM. The list that follows identifies the level of support or a reference to a clause where such support is specified. SAM permits each protocol to select the number of elements supported in several objects. The GPP implementation is stated below.

- GPP supports a maximum of 65 536 GPP devices. See 4.6 for the GPP domain specification. (See SAM, 3.4, Object Definition 1.)

- GPP supports up to 256 service delivery interfaces per GPP device. (See SAM, 3.6, Object Definition 2.)

- Each GPP device implements the SCSI device combined model. (See SAM, 3.6, Object Definition 3.)

- GPP supports one Initiator identifier per Initiator. GPP supports one Target identifier per Target. The Initiator identifier and the Target identifier are the same value in each GPP device. In GPP, these identifiers are 8 bytes long. The mode of operation of a GPP device in a Task provides the additional uniqueness to distinguish function. See 4.6, item 12. (See SAM, 3.6.1 and 3.6.2, Object Definitions 4 and 5.)

- GPP permits 1 to 256 Logical Units per Target. (See SAM, 3.6.3, Object Definition 6.)

- GPP supports Tasks with tags that permits an implementation level of concurrent Task execution in a logical unit. GPP supports up to 256 Tasks per Initiator per Logical Unit. (See SAM, 3.6.3, Object Definition 7.)

- GPP supports the procedural model for Tasks and Task management functions (see SAM, clauses 3.7, 4, and 5). See clause 8.

- If an error is detected outside the bounds of an active Task, the Logical Unit uses an asynchronous event reporting (AER) protocol called asynchronous event notification (AEN) to transfer appropriate sense data to affected Initiators that results in an ACA. The protocol for exiting the ACA is similar to the preceding example. This protocol conforms to the requirements of the optional asynchronous event reporting in SAM, clause 4.6.4.1.

- GPP supports the transission of the six Task Set management function identified in SAM, clause 5. See clause 7.

- Support for Task Set management operations in SAM, clauses 5 and 6, is the responsibility of each Logical Unit and is not specified in GPP.

4.6 Logical system model

A GPP logical system consists of up to seventeen (17) items. Items 1 through 13 are mandatory; items 14 through 17 are optional. Figure 4 shows these mandatory requirements and the defaults as they apply to an Initiator, a Target, and the path(s) between them. The numbers on the left side of Figure 4 refer to the item numbers in the model definition below.

1 A logical system consists of two logical elements with a minimum of one GPP port each and one GPP interface connecting them. The maximum number of logical elements is 65 536. See SAM, 3.4 and 3.6.)

2 Each GPP port is capable of operating in Initiator role on each GPP interface. (See SAM, 3.6.1.)

3 Each GPP port is capable of operating in Target role on each GPP interface. (See SAM, 3.6.2.)

4 A logical element that principally starts Tasks is called an Initiator (see SAM, 3.6). An Initiator controls one or more GPP ports per item 2. The same ports are used for Target role per item 3.

5 A logical element that principally receives Tasks and has its Logical Units execute them is called a Target (see SAM, 3.6). A Target controls one or more GPP ports per item 3. The same ports are used for Initiator role per item 2.

NOTE — The names given to the logical elements in GPP do not prevent any logical element from using all functions of GPP. The word "principally" in Items 5 and 6 implies this. Thus, a copy manager, acting principally as a Logical Unit, may act as an Initiator and use all defined functions of the commands and the logical system to perform a copy operation. Logical units also use AEN through Target ports in Initiator role.

6 The Initiator role port and Target role port in Items 2 through 5 above, attach to the same service delivery subsystem and are active in their respective roles during a connection; each port may reverse roles for each connection.

NOTE — For a full duplex port, the port may be in both roles at once. The connections may be logical rather than physical, as in some classes of service for Fibre Channel.

7 Each port has a minimum of one GPP address unique to the GPP interface to which it attaches.

NOTE — See the annexes for mapping GPP addresses to the physical address of an service delivery subsystem.

8 Each GPP port is assigned a port number by its controlling logical element. The port number is unique within each logical element.

9 Each Target has one or more Logical Units each identified by a unique Logical Unit number (LUN). The first LUN shall be 00h. Additional Logical Units may be any LUN value in the range of 01h to FFh. Each logical unit shall be referenced by the same Logical Unit number from each Target port (see SAM, 3.6.2).

NOTE — It is recommended that LUNs be assigned contiguously starting at 00h.

10 The extent of a logical system, for a Target, is the set of all Initiator/Initiator role port combinations attached, through one or more GPP interfaces, to the Target and from which an initial connection has been made. For an Initiator, the extent of a logical system is the set of all Logical Units to which an initial connection has been made on one or more GPP interfaces.

The extent of the logical system for an Initiator and a Target that are reachable by each other may not be the same. Multiple ported GPP devices are not required to be symmetrically attached to other devices. Reachable is defined as capable of a connect.

11 All GPP communication between Initiators and Targets shall be in the form of Information Packets, except primitive functions as defined for each service delivery subsystem.

12 Each logical element is assigned a world-wide name. The world-wide name shall be unique in the logical system. The world-wide name assigned to a set of Initiator mode ports is called WWNi. The world-wide name assigned to a set of Target mode ports is called WWNt (see SAM, 3.6.1 and 3.6.2).

An identifier consisting of a WWNi, an Initiator port number, an Initiator GPP address, a Target GPP address, a Target port number, a WWNt and a LUN, defines a path when the relationship is established as the result of a connect started by an Initiator to a Logical Unit. This is called an implicitly named path. The port numbers are transferred as part of each complete Task.

No path exists between a Logical Unit and an Initiator unless the Logical Unit is the object of a connect started by the Initiator to the Logical Unit.

An implicitly named path is in an ungrouped state. When a path is in the ungrouped state, each Task is limited to operation on the path where the connect was made. This is called single path mode.

Each Initiator connecting with a Target on any port must be considered as attached to a unique Initiator until the transfer of a WWNi from the Initiator occurs. Each Target role port connecting with a Initiator on any port must

be considered as attached to a unique Target until the transfer of a WWNt from the Target occurs. The transfer of world wide names is accomplished with messages.

13 For Initiators that can reach a Logical Unit (see Item 10), access may be restricted to certain items or extents within that Logical Unit, based on device class. Also, access may be temporarily restricted for a complete Logical Unit. These assignment functions provide the lowest level of restriction to information within a Logical Unit.

Any Logical Unit is initially available to receive Tasks from any Initiator attached to its Target. This default use state of each logical unit is called unassigned. This unassigned state exists whether the path is explicitly named, implicitly named, and whether for explicitly named paths, the path is grouped or ungrouped.

Access may be restricted to less than all Initiators in the logical system by using assignment. The functions of assignment are:

1 assign a Logical Unit to an implicitly named path

2 assign a Logical Unit to an explicitly named path which is in the ungrouped state

3 assign a Logical Unit to a path group (see item 16) (optional)

4 extend assignment of a Logical Unit to another established path group (see item 16) (optional).

The inverse of assign is unassign. The use state of a Logical Unit may be set to unassigned for any path to which assignment currently exists. The request to unassign a Logical Unit may be received on any path for which assignment currently exists. Assignment may be transferred from one Initiator to another without passing through a state where no assignment exists (see Item 17).

14 After the transfer of the WWNi and WWNt between the logical elements (i.e., an Initiator and a Target), the implicitly named path becomes an explicitly named path. An Initiator connects with and transfers its WWNi to each Logical Unit with which it needs to define an explicitly named path. The WWNt is transferred in response. The Logical Unit must be valid for the Target (i.e., connected or supported). The Logical Unit need not be ready or installed (e.g., powered off but cabled or not cabled). The Initiator is not required to transfer its WWNi on all available physical paths between the Initiator and the LUN, but only on those paths which it intends to use for additional GPP functions defined in Items 15 through 17. Once paths are named, the Target can distinguish which paths belong to the same Initiator and each Initiator can distinguish unique Logical Units.

An explicitly named path is initially in an ungrouped state. Tasks are handled in the same manner as an implicitly named path unless additional agreements are reached between the Initiator and the Target as defined in the items below.

15 An identifier, consisting of a WWNi , a WWNt, and a LUN represents one Logical Path. A Logical Path consists of a set of one or more explicitly named paths or exactly one implicitly named path. For implicitly named paths, the Initiator GPP address and port number and the Target GPP address and port number substitute for the WWNi and WWNt, respectively. The paths in a Logical Path are initially in an ungrouped state.

16 A set of ungrouped explicitly named paths in a Logical Path is established as a cooperating path group using any one of the paths in the Logical Path. This set of cooperating paths in a Logical Path is called a path group. Establishing a path group is the mechanism for enabling multiple path operations in a logical system. Including a path in a path group does not restrict access to the Logical Unit from any other path.

The inverse of establishing a path group is to disband a path group. A path group may be explicitly disbanded by the Initiator using any one path in the path group. A path group is implicitly disbanded when the last path is removed from an established path group using a remove path function.

When establishing a path group, the Initiator identifies where status leading to an auto-contingent allegiance is reported. Any status which does not result in auto-contingent allegiance may be sent over any path in the path group. The choices are:

- single path status mode
- multiple path status mode.

The status mode may be altered by disbanding the path group and establishing the path group with the alternate choice.

A path may be added to an established path group. A path may be removed from an established path group.

Once a path group is established, the extended functions of assignment and multiple path operation may be used (see Item 17).

17 Controlled access is the name given to the function which can break an assignment if some error or failure occurs in an Initiator which currently has assignment. Breaking assignment may be temporary or permanent, but it shall be controlled.

Controlled access permits access outside the bounds of assigned path groups. The mechanism to prevent deliberate or accidental loss of assignment protection is the control access function, enabled by a password and checked by the affected Logical Unit. The default state for a path group is that no password exists and controlled access is disabled.

Control access performs three functions:

1 Establish a password in a Logical Unit for a path group.

2 General unassign.

3 Request temporary unassignment.

A password is established for a path group by an Initiator having assignment. The password is not reported by a Logical Unit on any path. The Logical Unit checks its established password for the path group, if any, against the password supplied for a control access function being requested from an Initiator not having assignment.

If the Logical Unit has an established password for the path group and it matches the password with the control access request, the control access function and any linked commands for the associated Task are executed, if possible. One mechanism by which the unassigned Initiator may acquire the correct password is specified in GPP.

The password persists until the next power-on cycle or the Target or Logical Unit is reset (e.g., an event on the physical interface or a Task Management function).

NOTE — The processor command set (see SSC) between Initiators provides one mechanism to transfer the password between Initiators. The necessary functions to use the processor command set are currently required in GPP to support asynchronous event notification.

Figure 4 shows the relationship of these requirements and the internal states established, at minimum in each Initiator and each Target. The numbers on the left side of the figure refer to the 17 item numbers in the model above.

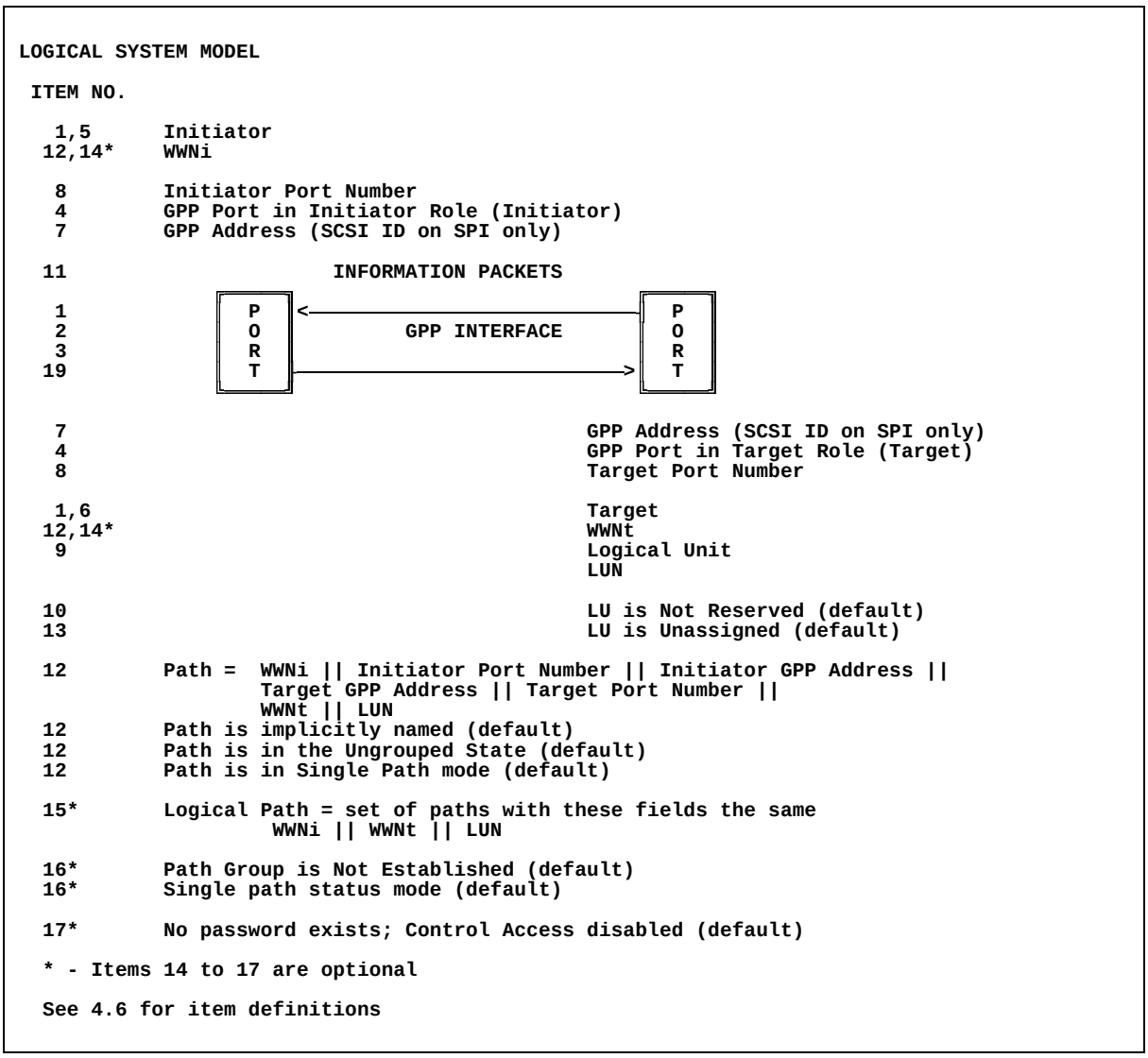


Figure 4 - Graphical representation of a logical system

4.7 Mandatory requirements

Mandatory requirements for logical operation for Initiators and Logical Units include:

1 All GPP ports shall implement Initiator role and Target role (i.e., dynamic switching of the port role).

2 Logical units shall implement untagged Tasks. Tagged Task support is optional.

3 All mandatory functions in the Information Packet interface control prefix shall be implemented.

4 All messages identified as mandatory shall be implemented, including service delivery subsystem specific messages when a port of that type is implemented.

5 An Initiator shall, in its Target role, provide a Logical Unit (LUN = 00h) that declares itself as a processor device class in response to an INQUIRY command and support the AEN version of the SEND command (See SSC). More then one Logical Unit may be provided in the same or different device classes.

6 All Initiators shall respond correctly to AEN. All Targets shall implement AEN with the default enabled and identified as not changeable in the mode pages.

7 All Targets shall implement Initiator mode on each port to select Initiators, to acquire the INQUIRY data to determine the principal Initiators, and to support AEN. Events not occurring in the bounds of a Task shall be reported when they occur using AEN.

5 GPP Tasks and Task management

This clause describes Tasks and Task management from the point of view of the protocol implementation. SAM describes tasks from the point of view of teh application client. Tasks and the components of Tasks are covered and then Task management is covered.

5.1 Tasks

Consider the simplified system shown in figure 5 where an Initiator and Target communicate on a single GPP interface to execute a Task. GPP requires that both the Initiator and the Logical Unit involved in a Task retain certain status about the operation of each Task and the ports with which it communicates.

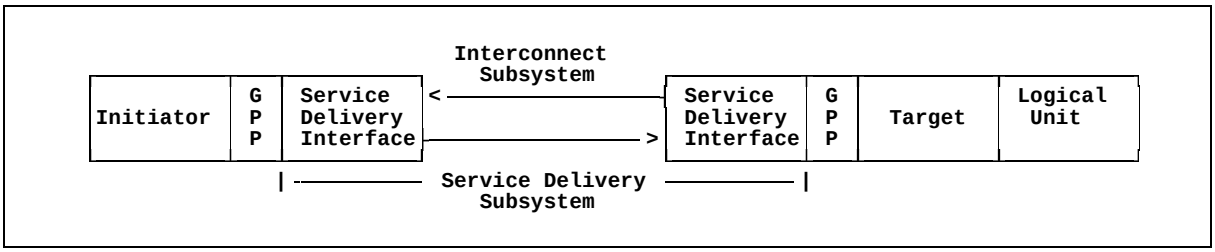


Figure 5 - Simplified GPP system

5.1.1 Parameter requirements (mandatory)

For logical elements, there are several kinds of parameters that each shall maintain:

1 a list of other logical elements that have been reached by this logical element.

2 a list of the ports reached in each other logical element and their GPP interface service parameters (e.g., for SPI, the synchronous data transfer negotiation agreement between the ports).

3 a list of the Logical Units in each other logical element for which a connect has been made. There shall be a minimum of one Logical Unit in each logical element.

4 Path names and path group agreements, assignment, and passwords for all Logical Units, including the no agreement states of these. Each Logical Unit shall maintain its current assignment status.

5 A Task set in an Initiator for which an initial connection has not been made and the associated information packet(s). The Initiator Task set may be empty.

6 A Task set and the state of each Task in both the Initiator and the Target for which an initial connection has been made. The state of each Task requires several elements:

- which Information Packets have been transferred,
- which Information Packets have been received,
- the logical order of the Information Packets,
- which ILEs in an Information Packet have been processed,
- a status field to hold the status for the Task and for each command of the Task,
- an indication of the current state of data transfer for each command, if any,
- an area to receive or prepare sense data for the Task should it end with ACA, and

- a means to tie the sense data to the correct portion the Task.

7 A Information Packet buffer to be used for holding Information Packets. The Information Packets may not be associated with a Task until later or they may be associated with a Task, for recovery on the interface, or the buffer may be used prior to establishing a new Task.

8 As the ILEs for a Task are being processed between the Initiator and the Logical Unit these parameters shall be updated. The update for received Information Packets is made when the Information Packet is processed; this may not be at the time it is received. The update for Information Packets sent is made as each information packet is transferred. The state of a Task may appear different at the Initiator and at the Logical Unit as a result of the potential difference in time for Task updates. The Logical Unit state of a Task takes precedence.

The logical element that receives an Information Packet maintains the true state of the Task. A transmitted and correctly received Information Packet from an Initiator shall not be considered processed by the receiving logical element until a positive indication is received back at the sender (e.g., status for a command or a request for more Information Packets with the TASK WAITING message).

5.1.2 Task sense data (mandatory)

Each Logical Unit shall maintain the status of that Logical Unit and shall maintain the minimum data as required by the device class and the device type implemented. This minimum data is called sense data. The format of the mandatory portion is specified with the REQUEST SENSE command of the SPC standard.

Sense data shall be kept current with the state of the Logical Unit and the Target. As events occur, the Logical Unit and the Target shall update the sense data. The Logical Unit shall be prepared to transmit its sense data as a response to a REQUEST SENSE command or as autosense for any status leading to an ACA.

5.1.3 GPP identification (mandatory)

There are three levels of logical addresses within GPP:

- the Initiator and Target
- the Logical Unit.

Associated with each port of a logical element, is a GPP address for each port and an internally assigned port number. These values are not new levels in the hierarchy of names. They form an alias identification for the logical element unique to the service delivery interface. Since the GPP address may not be not unique in the logical system, the GPP address alone cannot be used to establish multiple port operations.

5.1.3.1 Initiator identification (mandatory)

The Initiator identification (WWNi) is a world-wide name assigned to each Initiator in a logical system (see 4.6). By communicating this value to each Logical Unit, each Logical Unit can determine which physical paths belong to the same Initiator. Transfer of the WWNi to another logical element is a required step in enabling multiple path operations.

5.1.3.2 Target identification (mandatory)

The Target identification (WWNt) is a world-wide name assigned to each Target in a logical system (see 4.6). By communicating this value to each Initiator, the Initiator can determine which physical paths belong to the same Target. Transfer of the WWNt to another logical element is a required step in enabling multiple path operations.

5.1.3.3 GPP address for a port (mandatory)

A GPP port shall respond to at least one GPP address on one service delivery interface.

5.1.3.4 GPP port number (mandatory)

Each logical element shall assign an internally unique value to each port to a service delivery interface it controls. GPP limits the port number to a value in the range of 00h to FFh. This value is used in the Information Packet control prefix to identify the source port for each new Task.

5.1.3.5 Logical unit identification (mandatory)

Each Target shall have a minimum of one Logical Unit. LUNs may not be assigned contiguously through the maximum number of Logical Units supported by the Target. The maximum of Logical Units is 256 per Target.

Since each Initiator implements a target mode function as a processor class device, each Initiator, in its target mode, shall have at least one Logical Unit and that Logical Unit's LUN shall be zero.

NOTE — A logical element can determine which Logical Units each other logical element implements by sending an INQUIRY command for each LUN and examining the returned peripheral qualifier and peripheral device type fields.

Each Logical Unit shall be uniquely identified by WWNt || LUN (see SAM, 5.6.3).

5.1.4 Commands (mandatory)

A command consists of a CDB and, in some instances, command parameter data. Some ILEs are received before they are actively used in a Task. Three terms commonly used with commands are receipt, acceptance, and execution. The sense of these words for GPP is specified below.

- Receipt of a command is not the same as acceptance of a command. Receipt refers only to successful transfer of a command in an Information Packet. The command may, as part of its Task, go to the Task Set without interpretation in either the dormant state or the enabled state (see SAM, clause 6).

- Acceptance of a command means that the Task isin the enabled state, any prior command in the Task has completed successfully, all fields are valid for the implementation, and the current conditions in the Logical Unit permit attempting execution. A received command may be accepted or rejected.

  Command acceptance may not be indicated to the Initiator. The result of attempting to execute the command is reported in status which is the indication to the Initiator of whether the command has been accepted or rejected. Command acceptance may be implied by requests for additional Information Packets for the Task. (See the TASK WAITING message.)

- Execution applies only to commands that have been received and accepted and follow the Task Set management requirements of SAM (see SAM, clause 6). A command may be received and accepted and then cleared before execution has begun (see SAM, 6.5). The implemented procedure for the specific command is performed to the best ability of the Logical Unit. The command may terminate successfully, terminate with error, or be cleared, even if execution has begun (see SAM, clause 6).

A command which is received, accepted, executed, and completes successfully causes status and an appropriate command completion message to be prepared for transmission to the Initiator in an Information Packet. The status ILE and a command completion message ILE may be sent with other ILEs for the Task if additional ILEs are present in the Logical Unit for the Task.

A command which is received, accepted, and, after it execution begins, terminates with an error detected by the Logical Unit causes status to be prepared, sense data to be prepared, and an ACA to be established. An Information Packet is sent to inform the Initiator of the error which may contain sense data included as an autosense ILE. See 5.1.5 and 5.1.7. The final sequence of ILEs is:

- a status ILE indicating CHECK CONDITION or TASK TERMINATED,
- an autosense ILE (optional), and
- a message ILE indicating TASK COMPLETE.

A command which is received and rejected causes status to be prepared, sense data to be prepared, an ACA to be established (see SAM, 4.6.1). The Information Packet is sent to inform the Initiator of the error. A Task or any commands for the Task that are cleared receives no direct status report. A unit attention condition may be established by the Logical Unit for the Initiator (see SAM, 4.6.5).

GPP permits adding one new attribute, supervisor type, to Tasks. Supervisor type Tasks permit commands and messages identified as requiring supervisor mode to be executed. If the Logical Unit is not permitted to execute supervisory functions in a Task, any functions so identified shall not be executed and the Task terminated when the first one is encountered in an Information Packet. Such functions normally affect access to a Logical Unit. See the appropriate command set standards for commands that are identified as supervisory type commands. Messages that require a supervisory type Task are identified with the message description.

An Initiator indicates in the interface control prefix of an Information Packet when supervisor functions are permitted in each Task. The Initiator supervisory function control provides each Initiator an opportunity to control the content of Tasks which can affect availability and data reliability of attached Logical Units. Any command not defined as supervisor type may be executed in any Task. Any message not defined as supervisor type may be executed in any Task.

5.1.5 Status (mandatory)

Status is supported as specified in SAM, 4.2.

5.1.6 Asynchronous Event Notification (mandatory)

Support for asynchronous event notification (AEN) is mandatory in both Initiators and Targets. An asynchronous event is one which occurs in a Logical Unit that affects the operation of the Logical Unit with its Initiators, but which does not occur as a result of an active Task. Support for AEN meets the specified protocol specified method for asynchronous event reporting (see SAM, 4.6.4.1).

NOTE — The unit attention condition is an instance where AEN may be used. Another example of an asynchronous event is medium removal in a sequential access device, a removable optical device, or a CD-ROM. A GPP device, which normally functions as a Target with Logical Units, uses initiator role to notify an Initiator of an asynchronous event.

Retrieval of the INQUIRY data by Targets shall be performed prior to the first use of AEN with an Initiator. A Logical Unit at LUN 00h that responds to an INQUIRY command as a processor device class shall be notified of asynchronous events using this protocol. Such devices are normally Initiators. AEN shall not be used with a Logical Unit that reports a device class other than processor (e.g., devices executing COPY commands). A Logical Unit with a LUN value greater than 00h that declares itself as a processor class device may not implement the AEN protocol.

Each Logical Unit that responds as processor device class at LUN 00h shall be issued a TEST UNIT READY command to determine that the device is ready to receive an AEN. A processor device class Logical Unit that returns CHECK CONDITION status for the TEST UNIT READY command without autosense may be issued a REQUEST SENSE command to retrieve the sense data. The CLEAR ACA message may be sent to terminate the ACA. A Logical Unit that declares itself a processor device class at LUN 00h that returns GOOD status for a TEST UNIT READY command shall accept a SEND command with an AEN bit of one. (See the SPC standard, SEND command.)

A SEND command with an AEN bit of one shall be interpreted as an asynchronous event report by an Initiator. The sense data identifying the condition being reported shall be returned as a command parameter data of the SEND command (see the SPC standard). An ACA condition shall automatically be established for the Logical Unit identified in the command parameter data.

An error condition or unit attention condition shall be reported to an Initiator once per occurrence of the event causing it and only on one path to each Initiator (see 5.1.7). Any path to the Initiator from the Logical Unit may be used for AEN. This protocol is not intended to be used while an H_C_L relationship or H_C_L_Q relationship exists with an Initiator that has an active Task.

The procedure to identify candidate Initiators through their Logical Unit at LUN 00h shall be executed once at each power on or reset (internal or externally generated) and may be repeated whenever a Target deems it appropriate or when an event occurs that may invalidate the current information.

5.1.7 Auto-Contingent Allegiance (mandatory)

Auto Contingent Allegiance is supported (see SAM, 4.6.1). GPP supports ACA with or without the transfer of autosense. The transfer of autosense is not a requirement of ACA. The transfer of autosense cannot be requested by the application client (see SAM, 4.6.1.1). The application client may set the ACA field in a CDB, but this does not require the transfer of autosense (see SAM, 4.1.2).

5.2 Task Set management (mandatory)

There are two methods for specifying the groups of Task (see SAM, 3.6.3, Object Definition (7)):

- tagged
- untagged.

GPP permits a Logical Unit to support either or both groups. The nature of the GPP makes it possible to receive and store one or more Information Packets for a Task before any Information Packet is processed. Figure 6 shows the relationship between the types of Tasks and the type of relationship required to support them.

```
Task or Task Management relationship

-  H_C relationship (task management only - mandatory)
   - H_C_x_y Task relationship (not an H_C relationship)
     - H_C_L (untagged or task management - mandatory)
     - H_C_L_Q (tagged or task management - optional)

- path (I_T_) is physical interconnect between two ports on one interface

- for single path mode H_I_T_C replaces H_C in each Task relationship name (mandatory)

- requires a task or task management function
  - made up of one or more Information Packets
    - may become a current Task any connection on
      any path in the eligible set of paths

- may have a tag

LEGEND:

  H   = Initiator
  C   = Target
  I   = Initiator GPP address
  T   = Target GPP Address
  x_y = L or L_Q
```

Figure 6 - Nexus and task relationships

Information Packets permit the transfer of more than one ILE per connection for an Task. ILEs from one or more Information Packets may be suspended in the Logical Unit just as they may be suspended in the Initiator. That is, the entire Initiator portion of a Task, if permitted by the Information Packet transfer agreement, may be transferred to the Logical Unit in one set of Information Packets and suspended in the Logical Unit. The Initiator follows the progress of the Task by processing status, logical data, command response data, autosense, and messages received from the Logical Unit in Information Packets. If additional Information Packets are required, the Initiator sends them to the Logical Unit at the appropriate time.

If there is an error which causes termination of a Task, all of the remaining unprocessed ILEs in any Information Packet from the Initiator for the Task are purged.

If either the Initiator or the Logical Unit has no space to store a new Information Packet, the logical element shall return a busy indication for the new Information Packet. This indication is not the same as a BUSY status. The form of this response is service delivery subsystem specific. The Information Packet is discarded by the receiving logical element. The Information Packet may be resent at a later time.

GPP does not support real-time detection of overlapped Tasks since Information Packets may arrive out of order on some service delivery subsystems for a Task that contains multiple Information Packets from the Initiator. It is the responsibility of the Initiator to not identify two or more uncompleted Tasks with the same identification. Results are unpredictable.

5.3 Task termination (mandatory)

Each Task terminated either normally (i.e., within the expected protocol) or abnormally. The response provided by GPP to the upper layer is specified below. Notice that an upper layer is in either an Initiator or a Target.

5.3.1 Normal Task termination

A Task completes normally when the Logical Unit returns GOOD status and an associated TASK COMPLETE message for the Task (see SAM, 4.2, 4.4, and 6.2).

5.3.2 Abnormal Task termination

A protocol error by either the Initiator or the Target causes a Task to be terminated. Specific instances are identified throughout SCSI-3 (e.g., see SAM 4.4 and 6.2).

5.3.2.1 Initiator-caused Task termination

An initiator may use any of the Task management functions specified in SAM to terminate a Task before normal completion. With a full-duplex service delivery subsystem, these Task management functions may not have the desired effect on all Tasks if any Information Packets are in transit when one of the Task management functions is processed (see SAM, clause 5).

5.3.2.2 Target-caused Task termination

A Logical Unit terminates a Task abnormally when it returns CHECK CONDITION status or TASK TERMINATED status for the Task. A Logical Unit or a Target may also abnormally terminate a Task with an unexpected disconnect event (see SAM, 6.2).

5.4 Assignment termination (mandatory)

Assignment is terminated when a specific request to unassign is processed for a Logical Unit. When a path group that has assignment is disbanded, assignment is terminated.

5.5 Path group operations

Path group operations, as defined in GPP, are always present. Only the type of path group operation used is pertinent. There is single path mode and multiple path mode. Along with multiple path operations is the choice single path or multiple path status reporting for certain errors.

5.5.1 Single path mode (mandatory)

Unless multiple port operations are explicitly enabled and permitted for a Task, a Task shall operate exclusively on the path where the initial connection was made. This is called single path mode and is the default for all logical elements. That is, all relationships are of the form H_I_T_C or H_I_T_C_x_y, unless specifically enabled by the two logical elements for multiple port operation.

NOTE — This is the only mode of operation for SIP. Therefore, this mode in GPP is equivalent to SIP operation at the Task level.

5.5.2 Multiple path mode (optional)

When multiple paths have been identified between two logical elements, a Logical Path results which identifies a set of reachable physical paths between the two logical elements. If a Logical Path is established as a path group and the Task is specifically enabled by the Initiator, multiple path operation is possible. An Initiator and a Logical Unit in this mode shall choose, independently, which path in an established path group to use to send an Information Packet to the other, except as specified in 5.5.3.

5.5.3 Single path status (mandatory)

Unless a path group is established between two logical elements, all Tasks operate in single path status mode because the associated relationship is an H_I_T_C or an H_I_T_C_x_y relationship. The default is for all status to be reported only on the path where the initial connection was made. When a path group is established, the Initiator decides on the method of communicating CHECK CONDITION status or TASK TERMINATED status for any Task started using the path group where multiple path operation is allowed.

NOTE — This is the only mode of operation for SIP. Therefore, this mode in GPP is equivalent to SIP operation at the Task level.

5.5.4 Multiple path status (optional)

Each path group has the single path status or multiple path status attribute set when the path group is established and it remains unchanged until the path group is disbanded. For each Task, the Initiator enables or disables multiple path operation during the initial connection on a path in an established path group. When both the path group and the Task have multiple path status mode enabled, the Initiator permits CHECK CONDITION status or TASK TERMINATED status to be returned on any path in the path group where the initial connection was made. When multiple path operation is enabled for a path group and a Task, any other status may be reported on any path in an established path group. The Logical Unit shall choose, independently, where to report termination status.

5.5.5 Path group disbanding (optional)

A path group, once established, is disbanded as a result of a specific request to disband the path group or when the last path in the path group is removed from the path group. See the SET WWN message.

6 GPP Information Packet structure

This clause specifies the format of and requirements for Information Packets for GPP. The following subclauses specify the layout, interface control prefix and interface logical element layout for an Information Packet.

6.1 Information Packet layout

An Information Packet is set of self-describing ILEs arranged in prescribed orders based on the logical content being transmitted. The ILEs are surrounded by appropriate interface control fields necessary to send the logical content of the packet on a physical transport layer (see table 1). The following are minimum requirements for Information Packets. Additional requirements follow.

- the maximum Information Packet length shall be 65 536 bytes.
- Information Packets shall be a multiple of 4 bytes in length.
- An Information Packet shall contain at least one ILE and shall not contain more than 255 ILEs.
- Each ILE shall be a multiple of four bytes in length.
- ILEs may have pad bytes added to construct ILEs that are a multiple of 4 bytes in length. The value of pad bytes in GPP shall be 00h.

Table 1 - Information Packet Structure

Byte	Bits 7 - 0
0 to m	**Interface Control Prefix**
m + 1 to n	**one to 255 ILEs**

If an Information Packet is received that is not a multiple of 4 bytes long or there are zero or greater than 255 ILEs in the Information Packet, the Information Packet shall be rejected and no other action is taken. The Logical Unit returns an Information Packet with an INVALID INFORMATION PACKET message ILE.

Figure 7 contains a list of the supported ILE types.

- **Message**
- **Command descriptor block**
- **Command parameter data**
- **Command response data**
- **Logical data**
- **Status**
- **Autosense**

Figure 7 - ILE Types

6.2 Interface control prefix

The interface control fields encapsulate the information content of the logical operations. The identify function for both Initiators and Logical Units uses the interface control prefix fields to define a Task identifier or for Task Management. Table 2 shows the interface control prefix fields. All reserved fields and bits shall be set to zero.

Table 2 - Interface Control Prefix Fields

Byte	Bits							
	7	6	5	4	3	2	1	0
0 - 1	(MSB) Remaining Packet Length (LSB)							
2	Packet Type							
3	Reserved							
4	MltPath	SuspMpth	EnbSpvr	CtrlILEs	DataILEs	Reserved		
5	LUN Valid	TaskGp	TaskTp			Reserved		
6	Initiator Port Number							
7 - 8	Original Initiator GPP Address							
9 - 10	Original Target GPP Address							
11	Target Port Number							
12	LUN							
13	Tag							
14 - 15	(MSB) Sender's Sequence Number (LSB)							

The remaining packet length field indicates the total number of remaining bytes in an Information Packet in bytes, including the remaining interface control prefix fields. The actual usable maximum packet length is the result of negotiation and may be further restricted by maximum burst length negotiations. The remaining packet length field

value shall be of the form ((N*4) - 2) and greater than or equal to 18 ((16 + 4) - 2) and less than or equal to 65 534 (65 536 - 2).

The packet type field identifies the general content of an Information Packet.

- A packet type code of 00h indicates an Information Packet to start a new Task or Task Management function. A Task Management function may contain only Packet Type 00h.

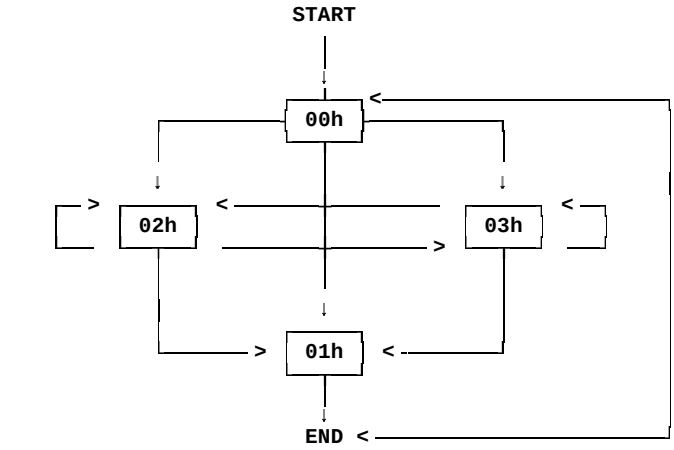
- A packet type code of 01h indicates a Logical Unit transmitted packet to terminate an Task.

- A packet type code of 02h indicates an intermediate Information Packet from an Initiator for an uncompleted Task established with an Information Packet of type 00h.

- A packet type code of 03h indicates an intermediate Information Packet from a Logical Unit for an uncompleted Task established with an Information Packet of type 00h.

- Packet type code values of 04h through FFh are reserved.

Figure 8 specifies the permissible flow of Packet Types for one Task.



Nexus Packet Type Ordering Rules

Current Type	Next Packet Type 0	1	2	3	END
Start	X				
0		X	X	X	X
1					X
2		X	X	X	
3		X	X	X	

LEGEND:

nnh - The numbers are hexadecimal packet type codes

Figure 8 - Packet Type flow for a nexus

A multiple path (MltPath) field value of one specifies that the Initiator considers the Logical Unit capable of multiple path operation. When the value is zero, the Initiator considers the path to be in single path mode. The setting has no effect on any other I/O process. This value shall remain unchanged in all Information Packets for the Task. The MltPath field indicates the Initiator state of the path. The Logical Unit shall confirm that its state for the path is the

same. If the Logical Unit state for the path is different, it shall not process the Information Packet and shall indicate the error by sending an Information Packet with the INVALID INFORMATION PACKET message. Accepting this field set to one is optional.

A suspend multiple path (SuspMpth) field value of one specifies that neither the Initiator nor the Logical Unit shall use multiple path operation for the Task. The effect is to form an H_I_T_C relationship from the H_C relationship. The SuspMpth field shall be interpreted for each Task. The setting has no effect on any other task in the Logical Unit. If the MltPath field is set to 0b, the SuspMpth field shall be set to 00h and the Logical Unit shall operate in single path mode for the Task. If the MltPath field is set to 1b and the SuspMpth field is set to 1b, the Logical Unit shall operate in single path mode for the Task. If the MltPath field is set to 1b and the SuspMpth field is set to 0b, the Logical Unit may operate in single path mode or multiple path mode for the Task. Accepting this field set to one is optional.

A enable supervisor function (EnbSpvr) field of one specifies that the Initiator has granted the Logical Unit the privilege of executing any valid supervisor function received during the task in which the enable supervisor command field is set to 1b. If the field value is zero and a supervisor function is detected in the task, the Logical Unit shall terminate the task with ACA. The sense key shall be set to ILLEGAL REQUEST and the additional sense code shall be set to SUPERVISOR FUNCTIONS NOT ENABLED. The enable supervisor function field setting is interpreted for each Task. The setting of this field shall have no effect on any other task. Accepting this field set to one is optional.

NOTE — When the MltPath, SuspMpth and EnbSpvr fields are all set to 0b, the method of operation for the task is logically equivalent to a SIP/SPI or SCSI-3 FCP Task on a single path.

A Control ILEs (CtrlILEs) field value of one specifies that the Information Packet contains one or more ILEs identified as a control type ILE in 6.3. Support of this field set to one is mandatory.

A Data ILEs (DataILEs) field value of one specifies that the Information Packet contains one or more ILEs identified as a data type ILE in 6.3. Support of this field set to one is mandatory.

A logical unit number valid (LUN Valid) field value of zero specifies that a Task Management function is directed to a Target and forms an H_C relationship. A LUN Valid field value of one specifies that the Information Packet is for a logical unit and forms an H_C_x_y relationship. Certain message ILEs can only be directed to a Target. The Logical Unit shall use this field, when 1b, as a valid field indicator for the LUN field. This value shall remain unchanged in all Information Packets for the Task. A Task Management function with a LUN Valid field value of zero shall operate in single path mode (i.e., if MltPath is set to 1b, SuspMpth shall be set to 0b). Support of this field set to one is mandatory.

The Task group (TaskGp) field value when one indicates that the Task is of the form H_C_L_Q. When the Task group field value is zero, the Task will be of the form H_C_L. The Logical Unit shall use this field value, when 1b, as a valid field indicator for the TaskTp and Tag fields. This value shall remain unchanged in all Information Packets for the Task. Support of this field set to one is mandatory.

The Task type (TaskTp) field value identifies the type of Task when the TaskTp field is set to 1b. When the TaskTp field is set to 0b, this field shall be set to 000b. Support of this field set to 001b shall be mandatory.

The type of Task is encoded into the TaskTp field as follows:

- 000b Simple (see SAM, 6.4.1)
- 001b ACA (see SAM, 6.4.4)
- 010b Ordered (see SAM, 6.4.2)
- 011b Reserved
- 100b Head of Task Set (see SAM, 6.4.3)
- 101b Reserved
- 110b Reserved
- 111b Reserved

The next six fields identify the path on which the initial connect was made for the Task. These fields shall not change in any Information Packet of any type transferred for the same Task once established as specified below. These fields, along with the LUN Valid, TaskGp and TaskTp fields identify the Task that is reestablished in any later connection on any allowable path for the Task.

The Initiator port number field identifies the internal port number assigned by the Initiator to the port where the initial connect was made using an Information Packet packet type code of 00h. This value shall remain unchanged in all Information Packets for the Task. Support of this field is mandatory.

The original Initiator GPP address field maps to the WWNi of the Initiator where the initial connect was made using an Information Packet packet type code of 00h. This value shall remain unchanged in all Information Packets for the Task. Support of this field is mandatory.

The original target GPP address field maps to the WWNt the GPP device, from the Initiator's perspective on the selected GPP bus, where the initial connect was made using an Information Packet type code of 00h. This value shall remain unchanged in all Information Packets for the Task. Support of this field is mandatory.

The Target port number field identifies the physical port number assigned by the Target to the Target GPP address that receives an Information Packet with a packet type code of 00h. This field shall be FFh when the packet type code is 00h. The Target supplies its unique port number value in any response Information Packet. Once the Initiator processes an Information Packet all subsequent Information Packets sent by an Initiator with packet type code 00h on the same physical path shall have the corresponding target port number placed in this field. The Logical Unit may receive multiple Information Packets during the connect with a target port number value of FFh. If the Logical Unit receives any Information Packet with packet type code 00h, other than as specified above, and the target port number does not match the internally assigned port number, the Logical Unit shall not process the Information Packet and shall indicate the error by sending an Information Packet with the INVALID INFORMATION PACKET message. Support of this field is mandatory.

The logical unit number (LUN) field specifies a logical unit number if the LUN Valid field value is set to one. This field is not meaningful when the LUN Valid field value is zero. This value shall remain unchanged in all Information Packets for the Task. Support of this field set to 00h is mandatory. Support of this field for values in the range 01h to FFh is optional.

The tag field is valid only when the TaskGp field value is 1. If the TaskGp field value is zero, this field shall be set to 00h. If the Logical Unit detects any value other than 00h when the TaskGp field value is zero, it shall indicate the error by sending an Information Packet with an INVALID INFORMATION PACKET message. This value shall remain unchanged in all Information Packets for a Task. Support of this field is mandatory.

The sender's sequence number is a value from 0 to 65 535 which specifies the sequence of an Information Packet relative to other Information Packets for the same Task from the same entity. The Initiator shall set this value to zero when the packet type code is 00h. It then increments the value by one for each subsequent Information Packet it sends for the same Task. The Logical Unit shall set the value of this field to zero on the first Information Packet sent in response to a Task. It shall then increment the counter by one for each subsequent Information Packet. If the value reaches 65 535, the next sequence number is 0 (i.e., the sequence counter wraps). Support of this field is mandatory.

If any reserved field is not zero, the receiver of the Information Packet shall not process the Information Packet further. It shall send an Information Packet with an INVALID INFORMATION PACKET message.

6.3 Interface Logical Elements (ILEs)

The ILEs are messages, command descriptor blocks (CDB), command parameter data, command response data, logical data, status, and autosense. A self-describing set of fields prefixes each ILE to permit the Information Packet content to be correctly identified, checked, and processed at the receiving GPP port (see table 3). There are two groups of ILEs:

- Message, command descriptor, status and autosense ILEs are control-type ILEs.

- Command parameter data, command response data and logical data ILEs are data-type ILEs.

Table 3 - Interface Logical Element (ILE)

Byte	Bits							
	7	6	5	4	3	2	1	0
0 - 1	(MSB) Remaining Element Length (LSB)							
2	Element Type							
3	Reserved						Pad Byte Count	
4 - n	Logical Element Bytes							

The remaining element length is a two-byte binary field. The value range is from 2 (4 - 2) to 65 518 ((65 536 - 16) - 2). The value is 2 plus the number of logical element bytes and pad bytes. If the element length value is 2 and the pad byte count field value is 00h, the ILE shall be ignored by the receiver of the Information Packet. This is not considered an error. When the remaining element length value is 2, the element type is not meaningful. When the remaining element length value is 2, and the pad byte count value is other than 00h, the receiver of the Information Packet shall send an Information Packet with an INVALID INFORMATION PACKET message.

The element type is valid when the remaining element length field value is greater than 2. The valid values for the element type field are defined in table 4. The Send and Receive columns in the table represent the operating mode of the port transmitting or receiving an Information Packet (i.e., Initiator or Target) containing an ILE of the type in the ILE column. If an element type value received is other than those defined in table 4 in the receiver column, the receiver of the Information Packet shall send an Information Packet with an INVALID INFORMATION PACKET message.

Table 4 - Element Type Codes

Element Type Value	ILE	Send Mode	Receive Mode
0h	Message	Initiator/Target	Initiator/Target
1h	Command Descriptor Block	Initiator	Target
2h	Command Parameter Data	Initiator	Target
3h	Command Response Data	Target	Initiator
4h	Logical Data	Initiator/Target	Initiator/Target
5h	Status	Target	Initiator
6h	Autosense	Target	Initiator
7h - FFh	Reserved	N/A	N/A

The pad byte count field indicates the number of pad bytes added at the end of an ILEs to construct an ILE that is a multiple of 4 bytes in length. Valid values are 00b, 01b, 10b, and 11b that represent 0, 1, 2, and 3, respectively, pad bytes appended to the ILE. An ILE with a logical element byte count that is an even multiple of four shall have the pad byte count value set to 00b.

The logical element bytes field is a 0 to 65 516 byte variable length field composed of the information about the element type declared in the element type field. The maximum length which may be used for this field may vary by the physical interface implemented and the currently negotiated maximum Information Packet length. The maximum length for any one ILE shall be limited by the requirements of the currently negotiated maximum Information Packet length. The current maximum ILE length is calculated as current maximum Information Packet length - 20.

Each ILE shall be rounded up to the nearest multiple of 4 bytes by adding from zero to three pad bytes. The value of each pad byte shall be set to 00h by the sender of an Information Packet. The value in these bytes are not meaningful at the receiver and shall not be checked.

If any reserved field is not zero, the receiver of the Information Packet shall not process the Information Packet further. It shall send an Information Packet with an INVALID INFORMATION PACKET message.

6.3.1 Message ILE

A message ILE shall contain one complete message. A multiple byte message shall not be split into bursts. See Table 4 for the element type field value for this ILE. See clause 7 for messages specified for GPP.

NOTE — In SIP, the logical element bytes in this ILE are transferred in the MESSAGE IN and MESSAGE OUT phases.

6.3.2 Command descriptor block ILE

A command descriptor block ILE shall contain one command descriptor block as described in the appropriate SCSI-3 standards. See Table 4 for the element type field value for this ILE. Any associated command parameter data is transferred in its own ILE. A command descriptor block shall not be split into bursts.

NOTE — In SIP, the logical element bytes in this ILE are transferred in the COMMAND phase.

6.3.3 Command parameter data ILE

The command parameter data ILE is a burst of data transferred for a command which is sent as an addendum for some CDBs. See Table 4 for the element type field value for this ILE. The CDB identifies the total length of the command parameter data implicitly or in the parameter length field. Each command parameter data ILE shall contain parameter data for only one command.

NOTE — In SIP, the logical element bytes in this ILE are transferred in the DATA OUT phase without any special differentiation of function in the protocol.

6.3.4 Command response data ILE

The command response data ILE is a burst of data transferred from a Logical Unit to an Initiator for a command that is not taken from the logical data of the LUN (i.e., not read/write type data). See Table 4 for the element type field value for this ILE. Sense data and MODE SENSE pages are examples of command response data. The CDB identifies the maximum length of the command response data implicitly or in the allocation length field. Each command response data ILE shall contain response data for only one command.

NOTE — In SIP, the logical element bytes in this ILE are transferred in the DATA IN phase without any special differentiation of function in the protocol.

6.3.5 Logical data ILE

A logical data ILE consists of a burst of data for the primary function of a logical unit requested for transfer between an Initiator and a Logical Unit. See Table 4 for the element type field value for this ILE. Data bytes written or read as logical blocks for a disk Logical Unit or data bytes to be processed for printing on a page are examples of logical data. Each logical data ILE shall contain data for only one command.

NOTE — In SIP, the logical element bytes in this ILE are transferred in the DATA IN and DATA OUT phases without any special differentiation of function in the protocol.

6.3.6 Status ILE

The status ILE contains a status value as defined in the appropriate SCSI-3 standard. Status shall not e split into bursts. See Table 4 for the element type field value for this ILE. Each Status ILE shall contain only one status code value.

NOTE — In SIP, the logical element bytes in this ILE are transferred in the STATUS phase.

6.3.7 Autosense ILE

The autosense ILE is a burst of data sense data transferred for an I/O process which terminates with ACA. See Table 4 for the element type field value for this ILE.Each autosense ILE shall contain data to report one event. An ACA remains in effect until terminated by the Initiator.

NOTE — In SIP, the logical element bytes in this ILE are transferred in the DATA IN phase without any special differentiation of function in the protocol in response to a REQUEST SENSE command or in a SEND command used for AEN.

7 GPP task control

This clause specifies the mandatory message ILEs required with the Generic Packetized Protocol. Table 5 identifies the messages and the requirements for implementation by Initiators and Targets.

Table 5 - Primary message codes

Code	Initiator S/R	Target S/R	Message Name	Direction	Singular
EXTD	O/M	M/M	ASSIGN	In/Out	Yes
EXTD	O/O	O/O	CONTROL ACCESS	In/Out	No
06h	O/N	N/M	ABORT TASK SET	Out	Yes
0Dh	O/N	N/M	ABORT TASK	Out	Yes
16h	M/N	N/M	CLEAR ACA	Out	Yes
0Eh	O/N	N/M	CLEAR TASK SET	Out	Yes
0Ch	O/N	N/M	TARGET RESET	Out	Yes
EXTD	M/M	M/M	EXTENDED MESSAGE REJECT	In/Out	Yes
EXTD	M/M	M/M	INVALID INFORMATION PACKET	In/Out	Yes
0Ah	N/M	M/N	LINKED COMMAND COMPLETE	In	No
0Bh	N/M	M/N	LINKED COMMAND COMPLETE (with FLAG)	In	No
EXTD	N/M	M/N	MODIFY DATA POSITION	In	Yes
EXTD	M/M	M/M	PACKET TRANSFER REQUEST	In/Out	Yes
EXTD	O/O	O/O	REPORT PATH STATUS	In/Out	Yes
EXTD	O/M	O/M	RESEND PREVIOUS INFORMATION PACKET	In/Out	Yes
EXTD	O/O	O/O	SET WWN	In/Out	Yes
00h	N/M	M/N	TASK COMPLETE	In	No
EXTD	O/M	O/M	TASK WAITING	In/Out	Yes
EXTD	O/N	N/M	TERMINATE TASK	Out	Yes
EXTD	O/M	M/M	UNASSIGN	In/Out	Yes
02h-FFh	-	-	Reserved if not listed above	-	-

KEY:

EXTD = Extended Message Code 01h
In = Target to Initiator
Out = Initiator to Target
M = Mandatory support
No = Ma be other ILEs in Packet
N = Not Sent or Not Received as appropriate by column
O = Optional support
S/R = Send / Receive
Yes = Only ILE in a Packet

In addition, each service delivery subsystem may require additional message ILE support to handle its specific needs beyond those specified in this clause. If such service delivery subsystem specific support is required, the messages shall be identified and their function shall be specified in the first annex for each service delivery subsystem.

No service delivery subsystem specific message shall use any message identification specified in this clause (i.e., the service delivery subsystem specific messages shall not define duplicate message codes to those specified in this clause.) Each independent service delivery subsystem may duplicate message codes used in other service delivery subsystems.

Singular messages are messages which shall be the only ILEs transferred in an Information Packet There shall be no other ILE types in the task. Any message not defined as a singular message may be combined with other ILEs for a task.

All service delivery subsystem specific messages shall be singular messages.

All messages shall follow the message structure specified in 7.1.

7.1 Message structure

Messages allow communication of information between an Initiator and a Target or a Logical Unit not contained in the other Information Packet ILEs. The logical element bytes of a message ILE may be one, two, or multiple bytes in length. Zero or more messages may be transferred along with other ILEs in a single Information Packet.

The Initiator and Logical Unit shall control the content of Information Packets when it sends any singular message identified in table 6. The column labeled "Singular" when it contains the word "YES" indicates the an Information Packet containing a singular message ILE shall contain only that ILE in the Information Packet. No other ILE of any type shall be in the same Information Packet. The assigned value ranges for messages and their lengths are defined in table 6.

All GPP devices acting as Initiators shall implement the mandatory messages tabulated in the "Initiator Mode" column of table 6 for Information Packet operations. All Logical Units shall implement the mandatory messages tabulated in the "Target Mode" column of table 6 for Information Packet operations.

Additional messages may be required for each different service delivery subsystem implemented. See the first normative annex for each service delivery subsystem.

One-byte, Two-byte, and extended message formats are defined. The first byte of the message ILE logical element bytes determines the format as defined in table 6.

Table 6 - Message Format Codes

Format Code	Message Element Byte Format
00h	One-Byte Message (TASK COMPLETE)
01h	Extended Messages
02-1Fh	One-Byte Messages
20-2Fh	Two-Byte Messages (Reserved)
30-FFh	Reserved

One-byte messages consist of a single byte where the value of the byte determines which function is to be performed. Two byte messages consist of a message code followed by one parameter byte. The message code determines which function is to be performed using the value in the parameter byte. GPP does not specify any two-byte messages and all message codes are reserved.

A value of 01h in the first byte of a message indicates the beginning of a multiple-byte extended message. The minimum number of bytes in an extended message is three. The maximum number of bytes in an extended message is 257 bytes. The extended message format is shown in table 7.

Table 7 - Extended Message Format

Byte	Value	Description
0	**01h**	**Extended Message Format Code**
1	**n**	**Extended Message Length**
2	**-**	**Extended Message Code (see Table 8)**
3 - n+1	**-**	**Extended Message Parameters**

The extended message length field specifies the length in bytes of the extended message code plus the extended message arguments to follow. The total length of a message is equal to the extended message length plus two. The minimum value of the extended message length field is 1; the maximum value is 255.

The function to be performed is determined by the value in the extended message code field. The extended message codes are listed in table 8. For extended messages, the extended message code field is followed by a variable length set of zero or more extended message parameter bytes. The extended message parameters are specified with each extended message description.

Table 8 - Extended Message Codes

Code Value	Name (Ordered by Extended Message Value)
0Ah	ASSIGN
0Bh	CONTROL ACCESS
04H	EXTENDED MESSAGE REJECT
06h	INVALID INFORMATION PACKET
00h	MODIFY DATA POSITION
07h	PACKET TRANSFER REQUEST
0Ch	REPORT PATH STATUS
05h	RESEND PREVIOUS INFORMATION PACKET
0Dh	SET WWN
09h	TASK WAITING
08h	TERMINATE TASK
0Eh	UNASSIGN
01h - FFh	Reserved if not listed (see Physical Interface Annexes)
	Name (Ordered by Extended Message Code)
00h	MODIFY DATA POSITION
04h	EXTENDED MESSAGE REJECT
05h	RESEND PREVIOUS INFORMATION PACKET
06h	INVALID INFORMATION PACKET
07h	PACKET TRANSFER REQUEST
08h	TERMINATE TASK
09h	TASK WAITING
0Ah	ASSIGN
0Bh	CONTROL ACCESS
0Ch	REPORT PATH STATUS
0Dh	SET WWN
0Eh	UNASSIGN

7.2 GPP primary messages

The messages specified in this clause are mandatory for all implementations of GPP as specified in table 6. Since all Initiators implement target mode and all Targets implement initiator mode any message which has a mandatory requirement for either mode shall be implemented in each GPP device for the corresponding mode.

Additional messages which are service delivery subsystem specific, if any, are found in the protocol specific annexes.

7.2.1 ABORT TASK

The ABORT TASK message is a singular message. An ABORT TASK message is sent from an Initiator to a Logical Unit to clear the nexus identified. This message shall not be sent from a Target to a Initiator. If the Logical Unit has already started execution of the task, execution shall be halted. The logical data contents of the logical unit may have been modified before the message was processed. In either case, any pending status or data for the task shall be cleared and no additional ILEs shall be sent to the Initiator.

It is not an error to issue this message to a logical unit that does not have a matching nexus in its task set.

Pending status, data, and commands for other tasks shall not be affected. Execution of other tasks shall not be aborted as a result of processing this message. Previously established conditions, including MODE SELECT parameters, assignments, and ACA shall not be changed by the ABORT TASK message.

The Information Packet containing an ABORT TASK message shall contain only one ILE, a message ILE.

7.2.2 ABORT TASK SET

The ABORT TASK SET message is a singular message. The ABORT TASK SET message is sent from an Initiator to a Target or Logical Unit. This message shall not be sent from a Target to a Initiator. When the message is sent using an H_C_x_y nexus to a Logical Unit, the Logical Unit shall clear the identified task plus any other tasks for the same Initiator. Pending data, status, and tasks for any other Initiator shall not be cleared as a result of processing this message.

It is not an error to issue this message to a logical unit that does not have any tasks for the Initiator.

When the message is sent using an H_C nexus to the Target, no status or message shall be sent for the current task and no other task shall be affected. Pending data, status, and other tasks for this Initiator or for any other Initiator shall not be cleared as a result of processing this message.

Previously established conditions, including MODE SELECT parameters, reservations, and ACA shall not be changed by the ABORT message.

7.2.3 ASSIGN

The ASSIGN message is a singular message. The ASSIGN message (table 9) requests that a logical unit be assigned to a specific path or path group. The message is also used to report the result of an assignment request or to report attempts to access a Logical Unit for which an Initiator does not have assignment. There are two protocols for this message:

- When the message is received as a request for assignment, the protocol for this message shall be one message in a single Information Packet in each direction from an Initiator to a Logical Unit and from that Logical Unit to the same Initiator.

- When the message is used to report an assignment conflict to an Initiator, the protocol for this message is a single message from the Logical Unit to the same Initiator. The task terminates with this message for this condition.

When the message is sent as an assignment request and all required conditions as specified below are met, the logical unit is assigned to a path or a path group and shall be accessible on that path or any path in that path group, respectively. If the assignment is requested for an implicitly named path, the Logical Unit shall be exclusively assigned to that one path.

NOTE — Assignment to an explicitly named path is logically equivalent to a unit reservation in SIP. That is, exactly one Initiator on exactly one Target port has access to the Logical Unit. Any other Initiator on any port is denied access.

Both Initiators and Logical Units shall keep track of assignment of Logical Units. Each shall communicate on the appropriate paths when assignment exists. The Logical Unit shall reject attempts for access from unassigned paths once assignment is established.

The ASSIGN message, when used to request assignment, specifies the path group on which a logical unit may operate. Two or more established path groups may share use of a Logical Unit to the exclusion of other paths or path groups. An implicitly named path or a path in the ungrouped state may hold assignment only for itself.

Additionally, the ASSIGN message may be used to assign only an extent of a block-type Logical Unit or to assign certain elements of a medium changer to a specific Initiator.

Any path holding an assignment through an established path group may be used to add assignment of other established path groups to the same assigned portion of the Logical Unit (the entire Logical Unit, an extent or an element as appropriate). The assigned portion of a Logical Unit may be assigned to multiple established path groups. An implicitly named path or an ungrouped path may be given assignment, but that path shall not be used to add assignment for any other paths or path groups.

Confirmation that assignment has been made is reported by returning the identical message content to the requester. If the assignment request cannot be processed, the Logical Unit indicates this to the requester by returning a special form of the ASSIGN message as specified below.

When assignment exists for a path group, a set of path groups, or for only one implicitly named path or ungrouped path, no Initiator on any path not in the assigned path group(s) or the single assigned implicitly named path or ungrouped path, respectively, shall gain access to the logical unit unless temporary assignment has been granted through the CONTROL ACCESS message. If an attempt is made to access a Logical Unit on an unassigned path, the Logical Unit shall send this message to the same Initiator using the special format specified below which prevents establishing an ACA with the unassigned Initiator.

NOTE — An ACA would violate the assignment by permitting the unauthorized initiator an opportunity to start an ACA task (possibly containing linked commands) which would violate the assignment.

Table 9 - ASSIGN message format

Byte	Value	Description
0	01h	Extended Message Format Code
1	0Eh or 0Eh + (12 * No. Extended Assignment Extended Descriptors)	Extended Message Length
2	0Ah	Extended Message Code
3	xxh	WWNi Length
4 - 11	xxh	(MSB) WWNi (LSB)
12	xxh	Assignment Type
13	xxh	Assignment Identification
14	00h	Reserved
15	xxh	Request Status
16 - 16 + n*12, n=0..20	xxh	Extended Assignment Descriptors 1 - 20 (12 bytes each)

When the assignment type field is set to 00h, the Extended Message length field shall be set to 0Eh. When the assignment type field is set to a value other than 00h, the Extended Message length field shall be set to 0Eh + (12 * number of extended assignment descriptors present). The ASSIGN message permits 0 to 20 extended assignment descriptors to be specified in the message, depending on assignment type.

When the assignment type field value is set to 00h, the WWNi length field shall have either a value of zero or 8. If the WWNi length field contains any other value the response message shall be returned with the request status field set to FEh. The request status field shall be set to 00h. The assignment identification field shall be set to 00h.

When the ASSIGN message is used to report the result of a successful assignment request, all fields shall contain the same value as in the original request message. If the assignment is not made, the value in the request status field in the response message shall be set to FFh to indicate that the assignment request failed. The WWNi field shall be the same value as in the original request.

When the ASSIGN message is used to report that an Initiator does not have current assignment to the Logical Unit, the request status field in the response message shall be set to FDh to indicate that an assignment conflict resulted from this nexus. The nexus shall be terminated with transfer of this message. The WWNi field, the WWNi, the assignment type, and assignment identification fields shall be ignored. The extended message length field value shall be 0Eh.

When the WWNi length field value is 08h, the WWNi field consists of an 8 byte hexadecimal value which uniquely names the path group for which assignment is to be made. This value shall be the WWNi of the Initiator.

If the WWNi length field is set to zero, the Initiator is requesting assignment on the path where the initial connection was made. The WWNi field shall be ignored. The Logical Unit shall return the same ASSIGN message to the Initiator

to indicate a successful processing of the assignment. If an ASSIGN message with the WWNi length field set to 8 is received on a path for which assignment does not exist, the request status field in the response message shall be set to FCh to indicate the error. If the WWNi length field is 8, the WWNi field matches the name of another established path group, and the path on which the message is received has assignment, the Logical Unit shall add assignment of the Logical Unit to the other established path group. It shall not be an error to receive this message on a path which currently has assignment and which requests assignment for another established path group for which assignment currently exists. The Logical Unit shall respond in the same manner as if the assignment did not exist.

If an ASSIGN message with the WWNi length field set to 8 is received on a path for which assignment exists and the WWNi field does not match the WWNi of another established path group in the Target. the request status field in the response message shall be set to FBh to indicate the error. All other fields shall remain unchanged in the response message. The assignment type field values identify the type of assignment requested:

- 00h Logical Unit assignment
- 01h extent assignment
- 02h element assignment
- 03h-FFh reserved

If the assignment type field is any reserved value, the message shall not be accepted and the request status field of the response message shall be set to FBh.

The assignment identification is meaningful only when the assignment type field is other than 00h. When the assignment type field value is 00h, this field shall be ignored. When the assignment type is other than 00h and the assignment is successful with a match on the assignment identification field value, the Logical Unit shall add the assignment identification field value and the assignment. When an ASSIGN message is processed the assignment identification value identifies which assignment is to be established.

The request status field shall be 00h on each assignment request. The request status field shall be 00h in a response message when the assignment is successful. Other valid values in this field in a response message are specified in this subclause.

The extended assignment descriptors are meaningful and a minimum of one descriptor shall be present when the assignment type is other than 00h. When the assignment type field value is 00h, this field shall not be present and the extended message length field value shall be 0Eh. When the assignment type is not 00h, the extended message length value set to 0Eh + (12 * number of extended assignment descriptors) in the message. The extended assignment descriptor formats are specified in the following subclauses.

7.2.3.1 Extent assignment

Extent assignment applies to device classes which have their medium divided into logical blocks (e.g., disks, write once device classes). Table 10 specifies the extended element descriptor used for extent assignment.

When the assignment type is set to 01h, and the extent reservation option is implemented, the Logical Unit shall process the assignment request as follows:

- If the number of extended element descriptors is larger than the Logical Unit supports, or the number to be added exceeds the total number supported if the message is executed, the Logical Unit shall reject this message, and send a response message with the request status set to F8h.
- The extended element descriptors shall be checked for valid logical block addresses. If any logical block address is invalid for this Logical Unit, the message shall be rejected with a request status of FAh. The extended element descriptors shall be checked for invalid logical block overlaps with other extended element descriptors in the message and if invalid overlaps are found, the command shall rejected with a request status of F9h.

- If the requested assignment does not conflict with any active or previously requested assignment, the logical blocks specified shall be assigned until superseded by another valid ASSIGN message from an Initiator within an assigned path group, or until released by an UNASSIGN message from Initiator within an assigned path group, by a TARGET RESET message from any assigned Initiator, or by a reset event.

- If the assignment request conflicts with an assignment already active the Logical Unit shall send a response message with the request status field set to F9h.

Table 10 - Extended element descriptor for extent assignment

Byte	Value	Description
0, Bits 7-2	000000b	Reserved
0, Bits 1-0	xxb	Assignment Class
1 - 3	xxh	Reserved
4 - 7	xxh	(MSB) Logical Block Address (LSB)
8 - 11	xxh	(MSB) Number of Blocks (LSB)

The classes of assignment within an extent type assignment are:

- 00h Read shared

- 01h Write exclusive

- 02h Read exclusive

- 03h Exclusive access

- 04h - FFh Reserved

A read shared assignment prevents all write-type operations to the assigned range of logical blocks by an Initiator in any path group including the Initiator making the assignment. Read-type operations are permitted by any Initiator in an assigned path group. Compatibility of this class of assignment with other classes of assignment is specified in figure 9.

A write exclusive assignment prevents all write-type operations by all Initiators in other path groups. Any Initiator within the path group where the assignment is made shall have write access to the extent. Read-type operations are permitted by any Initiator in an assigned path group. Compatibility of this class of assignment with other classes of assignment is specified in figure 9.

A read exclusive assignment Compatibility of this class of assignment with other classes of assignment is specified in figure 9.

An exclusive access assignment prevents all operations (read-type and write-type) by all Initiators in other path groups. Any Initiator within the path group where the assignment is made shall have read and write access to the extent. Compatibility of this class of assignment with other classes of assignment is specified in figure 9.

Assignment Class	00h	01h	02h	03h
00h Read shared	NO	YES	YES	YES
01h Write exclusive	YES	YES	NO	YES
02h Read exclusive	YES	NO	YES	YES
03h Exclusive access	YES	YES	YES	YES

LEGEND:
YES - this class of assignment conflict and is not permitted.
NO - This class of assignment does not conflict and is permitted.

Figure 9 - Assignment class conflict specification

Each extent assignment descriptor specifies a series of logical blocks beginning at the specified element address for the specified number of elements.

If the number of elements is zero, the assignment shall begin at the specified element address and continue through the last logical block address on the Logical Unit.

7.2.3.2 Element assignment

Element assignment applies to only to the medium changer device class. Table 11 specifies the extended element descriptor used for extent assignment. When the assignment type is set to 02h, and the element reservation option is implemented, the Logical Unit shall process the assignment request as follows:

- If the number of extended element descriptors is larger than the Logical Unit supports, or the number to be added exceeds the total number supported if the message is executed, the Logical Unit shall reject this message, and send a response message with the request status set to F8h.

- The extended element descriptors shall be checked for valid element addresses. If any element address is invalid for this Logical Unit, the message shall be rejected with a request status of FAh. The extended element descriptors shall be checked for invalid element overlaps with other extended element descriptors in the message and if invalid overlaps are found, the command shall rejected with a request status of F9h.

- If the requested assignment does not conflict with any active or previously requested assignment, the elements specified shall be assigned until superseded by another valid ASSIGN message from an Initiator within an assigned path group, or until released by an UNASSIGN message from Initiator within an assigned path group, by a TARGET RESET message from any assigned Initiator, or by a reset event.

- If the assignment request conflicts with an assignment already active the Logical Unit shall send a response message with the request status field set to F9h.

Table 11 - Extended element descriptor for element assignment

Byte	Value	Description
0, Bits 7-2	000000b	Reserved
0, Bits 1-0	xxb	Assignment Class
1 - 3	xxh	Reserved
4 - 7	xxh	(MSB) Element Address (LSB)
8 - 11	xxh	(MSB) Number of Elements (LSB)

The assignment class field is reserved for element assignments.

Each element assignment descriptor specifies a series of elements beginning at the specified element address for the specified number of elements.

If the number of elements is zero, the assignment shall begin at the specified element address and continue through the last element address on the Logical Unit. Element addresses not defined in a discontinuous assignment of element addresses shall be ignored and not assigned. This shall not be considered an error.

7.2.3 CLEAR ACA

The CLEAR ACA message is a singular message. The CLEAR ACA message is sent from an Initiator to a Logical Unit to terminate an ACA. An ACA ends on successful processing of the CLEAR ACA message.

The Information Packet containing a CLEAR ACA message shall contain only one ILE, a message ILE.

If a CLEAR ACA message is received by a Logical Unit when no ACA is active, the message shall not be rejected. The message shall be ignored.

7.2.4 CLEAR TASK SET

The CLEAR TASK SET message is a singular message. A CLEAR TASK SET message is sent from an Initiator to a Logical Unit to perform an action equivalent to receiving a series of ABORT messages from each Initiator. This message shall not be sent from a Target to a Initiator. All tasks, from all Initiators, for the specified logical unit shall be cleared. The logical data of the logical unit may have been altered by partially completed tasks.

All pending status and data for that logical unit for all Initiators shall be cleared. No status or message shall be sent for any of the tasks. A unit attention condition shall be generated for all other Initiators for which tasks were aborted for that logical unit. When reporting the unit attention condition the additional sense code shall be set to TASKS CLEARED BY ANOTHER INITIATOR. Previously established conditions, including MODE SELECT parameters,

reservations, and ACA shall not be changed by the CLEAR TASK SET message. The Information Packet containing a CLEAR TASK SET message shall contain only one ILE, a message ILE.

7.2.5 CONTROL ACCESS

The CONTROL ACCESS message (table 12)

1 establishes a password in a Logical Unit on an assigned path for a path group;

2 permits general unassign of a Logical Unit from an assigned path group;

3 allows an unassigned path that provides the correct password access to a logical unit for that task.

4 permits Initiators to transfer passwords to each other.

5 reports the status of the request to the requestor.

The protocol for this message shall be one message in one Information Packet in each direction from the Initiator to the Logical Unit and from the Logical Unit to the Initiator.

For a Logical Unit which is of the processor device class, the message may be processed as specified below. When no assignment exists and the EstabPwd field value is 02h, the message is processed to transfer a password between two cooperating Initiators.

Table 12 - CONTROL ACCESS message format

Byte	Value	Description
0	01h	Extended Message Format Code
1	0Eh	Extended Message Length
2	0Bh	Extended Message Code
3	08h	Password Length
4 - 11	xxh	(MSB) Password Name (LSB)
12	xxh	EstabPwd
13	xxh	GnlUnasn
14	xxh	ReqTAsgn
15	00h	Request Status

The CONTROL ACCESS message permits access to a Logical Unit outside the bounds of assigned path groups. The CONTROL ACCESS message, which provides a mechanism to prevent deliberate or accidental loss of assignment protection, is enabled by a password, and checked by the affected Logical Unit.

NOTE — A task containing a valid CONTROL ACCESS message requesting temporary assignment, followed by a CONTROL ACCESS message performing a general unassign, and then an ASSIGN message request for the new Initiator breaks all old assignments and transfers assignment of the Logical Unit to the new Initiator. This sequence of request messages and their responses permits continued operation on a different Initiator without loss of the function provided by the Logical Unit. Some tasks on the previously assigned paths or path groups may be aborted as a result of the general unassign function.

For a Logical Unit which is not of the processor device class, a password is established in a Logical Unit by an Initiator that has assignment. The password is not reported by the Logical Unit on any path. The Logical Unit checks its established password, if any, against password supplied in subsequent CONTROL ACCESS messages requesting temporary assignment from any Initiator not having assignment.

If the Logical Unit has an established password and it matches the password in the request message, the CONTROL ACCESS message is processed and a CONTROL ACCESS response message is sent to the requestor to confirm that access has been granted. The Initiator may continue the task with additional messages or commands.

NOTE — After a request for temporary assignment is granted, the Initiator may provide an ASSIGN message which will grant assignment to the once unassigned Initiator. The Initiator can then use normal tasks.

A status that would lead to ACA on the unassigned path having temporary assignment shall not be reported on the unassigned path, since that would grant the unassigned path permission to start a second task (first to retrieve the sense data and then to possibly link additional commands. The ACA is made to an assigned path whether functional or not. AEN may be used to report the condition on the selected assigned path.

If the message requests a password to be assigned, the message shall be accepted only on a path currently holding assignment. See the request status field below for reporting an assignment conflict. If the message requests a password to be transferred, the message shall be accepted on any path by a Logical Unit which is of the processor device class. The Password length field shall have a value of 8. See the request status field below for reporting invalid values.

The Password name field consists of an 8 byte hexadecimal value which provides a password for the Logical Unit. The contents of the field are valid only when the Password length field is set to 8.

The EstabPwd, field shall have only valid values of 00h, 01h, and 02h. See the request status field below for reporting invalid values.

- If the EstabPwd field is set to 00h, this function is not being invoked.

- When the EstabPwd field is other than zero, the GnlUnasn field and the ReqTAsgn field shall be set to zero.

- If the establish password (EstabPwd) field is set to 01h, establish a password in a Logical Unit. Only one password, the first password received on an assigned path, shall be the password for the Logical Unit. If a password exists for a Logical Unit and the password in the password name field is not equal to the established password, see the request status field below for reporting an attempt to establish a new password.

- If the establish password (EstabPwd) field is set to 02h, establish a password in another Initiator. This field value shall only be valid when received by a Logical Unit that declares itself a member of the processor device class in response to the INQUIRY command. Only one password, the first password received, from another Initiator, shall be the password retained for the Initiator sending the message. If a password exists for an Initiator and the password in the password name field is not equal to the established password, see the request status field below for reporting an attempt to establish a new password.

The GnlUnasn, and ReqTAsn fields shall have only valid values of zero and one. See the request status field below for reporting invalid values.

If the general unassign (GnlUnasn) field is set to one and no password has been established by a previous CONTROL ACCESS message or the password supplied matches the password in the Logical Unit, the Logical Unit removes assignment for all paths in any path group having assignment when the message was executed. If a password exists

for a Logical Unit and the password in the password name field is not equal to the established password, see the request status field below for reporting an attempt to bypass a valid password. If the GnlUnasn field is set to zero, this function is not being invoked. When the GnlUnasn field is other than zero, the EstabPwd field and the ReqTAsgn field shall be set to zero.

If the request temporary assignment (ReqTAsgn) bit is set to one, and the CONTROL ACCESS message is received from a path that does not have assignment and no password exists or the password matches the established password in the Logical Unit, the CONTROL ACCESS message and any commands for the task are executed to the extent possible. If a password exists for a Logical Unit and the password in the password name field is not equal to the established password, see the request status field below for reporting an attempt to bypass a valid password. If the ReqTAsgn field is set to zero, this function is not being invoked. When the ReqTAsgn field is other than zero, the EstabPwd field and the GnlUnasn field shall be set to zero.

When the CONTROL ACCESS message is a request, the request status field shall be set to zero. See the request status field below for reporting invalid values.

When the CONTROL ACCESS message is a response, the request status field shall have one of the values specified below:

1 A value of 00h indicates that the request was processed correctly by the Logical Unit It is not an error to attempt to establish the same password more than one time and on different paths. In this case, request status 00h is used if all other conditions are met.

2 A value of 01h indicates that more than one of the fields EstabPwd, GnlUnasn, or ReqTAsn, was set to one.

3 A value of 02h indicates that an in invalid value in the Password Length field.

4 A Value of 03h indicates an invalid value in the EstabPwd, GnlUnasn, or ReqTAsn fields.

5 A value of 04h indicates that the request status field was not zero in the request message.

6 A value or 05h indicates that a request to establish a passwords was received on a path that does not currently hold assignment. Access is denied. The task terminates.

7 A value of 06h indicates that an attempt was made to establish a second password of a different value. The new password is not accepted. The task terminates.

8 A value of 07h indicates an attempt to bypass a valid password.

9 Values 08h through FFh are reserved.

7.2.6 EXTENDED MESSAGE REJECT

The EXTENDED MESSAGE REJECT message is a singular message. The EXTENDED MESSAGE REJECT message (table 13) is sent from either the Initiator or Logical Unit to indicate that a portion of a message in a message ILE in the last Information Packet was inappropriate or has not been implemented.

The content of the extended message identifies the message ILE. The message indicates the reason for the rejection. The message also indicates the number of additional Information Packets not processed for the same task. This provides a logical interlock so that the receiver of the message can determine which message byte is in error.

The Information Packet containing an EXTENDED MESSAGE REJECT message shall contain only one ILE, amessage ILE.

Table 13 - EXTENDED MESSAGE REJECT message format

Byte	Value	Description
0	01h	Extended Message Format Code
1	0Ah	Extended Message Length
2	04h	Extended Message Code
3	xxh	ILE Position
4 - 5	xxxxh	(MSB) Byte Position in ILE (LSB)
6	00h	Reserved
7	xxh	Reject Reason Code
8 - 9	xxxxh	(MSB) No. of Information Packets Not Processed (LSB)
10 - 11	xxxxh	(MSB) Sender's Sequence No. of the First Information Packet Not Processed (LSB)

The ILE Position field value is the ordinal position of the message ILE within the Information Packet containing the error. The ILE Position is a value between 1 to 255.

The Byte Position in ILE is the byte position of the byte in error within the element bytes of the message in error. The value is 0 for the first message byte in the ILE. The range is 0 - 255.

The Reject Reason Code identifies the reason for sending the message. Allowable values for the Reject Reason Code are:

- A reject reason code of 00h indicates that the message code in message is not implemented.
- A reject reason code of 01h indicates that the message is inappropriate.
- A reject reason code of 02h indicates that the extended message length field in an extended message is invalid for the extended message code.
- A reject reason code of 03h indicates that a parameter byte within two-byte or extended messages is invalid.
- Reject reason codes 04h-FFh are reserved.

The Number of Information Packets Not Processed field contains a count of additional Information Packets (not counting the Information Packet in which the error was detected) for this task which have been discarded as a result of this error.

The Sender's Sequence No. of the First Information Packet Not Processed field identifies the Information Packet sequence number where the first Information Packet was discarded.

7.2.7 INVALID INFORMATION PACKET

The INVALID INFORMATION PACKET message is a singular message. The INVALID INFORMATION PACKET message (table 14) is sent from either the Initiator a Target, or a Logical Unit in an Information Packet to indicate that all or a portion of an Information Packet transferred was invalid. The format of the Information Packet does not conform to the rules for Information Packet construction. Errors in the content of the element bytes of ILEs is not pointed to using this message; see sense data.

The Information Packet containing an INVALID INFORMATION PACKET message shall contain only one ILE, a message ILE.

Table 14 - INVALID INFORMATION PACKET message format

Byte	Value	Description
0	01h	Extended Message Format Code
1	0Ah	Extended Message Length
2	06h	Extended Message Code
3	xxh	ILE Position
4 - 5	xxxxh	(MSB) Relative Position (LSB)
6 - 7	xxxxh	(MSB) No. of Information Packets Not Processed (LSB)
8 - 9	xxxxh	(MSB) Sender's Sequence No. of the First Information Packet Not Processed (LSB)
10 - 11	0000h	Reserved

The ILE Position field is the ordinal position of the ILE group within the Information Packet containing the error. The ILE Position is a value between 0 to 255. There may be more than one ILE in each Information Packet. A value of 0 in ILE Position indicates that the information control prefix fields are in error.

The Relative Position field identifies the byte position of the field or bit in error. The exact bit position is not provided. The value gives the relative position of the byte to the beginning of the ILE when the ILE Number value is 0 to 255. The value gives the relative position of the byte to the beginning of the Information Packet when the ILE Number value is zero.

The Number of Information Packets Not Processed field contains a count of additional Information Packets (not counting the Information Packet in which the error was detected) for this task which have been discarded as a result of this error.

The Sender's Sequence No. of the First Information Packet Not Processed field identifies the Information Packet sequence number where the first Information Packet was discarded.

7.2.8 LINKED COMMAND COMPLETE

The LINKED COMMAND COMPLETE message is sent from a Logical Unit to an Initiator to indicate that the execution of a linked command has completed and that status has been sent. This message shall not be sent from an Initiator to a Target. The next command for the task may come from an Information Packet already transferred or it may come from the Initiator in a new Information Packet.

7.2.9 LINKED COMMAND COMPLETE (WITH FLAG)

The LINKED COMMAND COMPLETE (WITH FLAG) message is sent from a Logical Unit to an Initiator to indicate that the execution of a linked command (with the flag bit set to one in the CDB) has completed and that status has been sent. This message shall not be sent from an Initiator to a Target. The next command for the task may come from an Information Packet already transferred or it may come from the Initiator in a new Information Packet.

7.2.10 MODIFY DATA POSITION

The MODIFY DATA POSITION message is a singular message. The MODIFY DATA POSITION message (table 15) requests that the signed argument be added (two's complement) to the value of the current logical block position in the Initiator. This message shall control or adjust the logical block data transfer position for an task.

The Information Packet containing a MODIFY DATA POSITION message shall contain only one ILE, a message ILE.

Table 15 - MODIFY DATA POSITION message format

Byte	Value	Description
0	01h	Extended Message Format Code
1	0Ah	Extended Message Length
2	00h	Extended Message Code
3	00h	Reserved
4 - 7	xxxxxxxxh	(MSB) Change in Relative Position (LSB)
8 - 9	xxxxh	(MSB) No. of Information Packets Not Processed (LSB)
10 - 11	xxxxh	(MSB) Sender's Sequence No. of the First Information Packet Not Processed (LSB)

If the change in relative position field value, in twos complement format, added to the current initiating controller results in a value less than zero or greater than the total length for the command, the Initiator shall transfer an Information Packet containing an ABORT TASK message and shall not process subsequent Information Packets for the task.

The Number of Information Packets Not Processed contains a count of additional Information Packets (not counting the current Information Packet in which the error was detected) for this task which have been discarded as a result of this error. The current Information Packet is the Sender's Sequence Number of the First Information Packet Not Processed value minus 1.

The Sender's Sequence Number of the First Information Packet Not Processed field identifies the Information Packet sequence number where the first Information Packet was discarded. If no Information Packets beyond the current Information Packet were discarded, the value in this field shall be the current Information Packet number plus one and the Number of Information Packets Not Processed field value shall be zero.

7.2.11 PACKET TRANSFER REQUEST

The PACKET TRANSFER REQUEST message is a singular message. A PACKET TRANSFER REQUEST (PTR) message (table 16) exchange shall be initiated by an GPP device whenever a previously arranged Information Packet transfer agreement may have become invalid or conditions in the GPP device warrant a renegotiation. Each GPP device shall retain at minimum one Information Packet per task for retransmission (i.e., the last Information Packet successfully transferred). Each GPP device shall support Information Packet lengths of 256 bytes or larger.

The Information Packet containing an PACKET TRANSFER REQUEST message shall contain only one ILE, a message ILE.

The PACKET TRANSFER REQUEST exchange may result in asymmetric values between the exchange participants. That is, the Initiator may establish fewer and shorter Information Packets than the Target. This asymmetry permits each unit to select the size of Information Packet it receives and the number of Information Packets it retains.

In the absence of an explicit packet offset count agreement for a value larger than one, the implied packet offset count agreement shall be 1 for both Initiators and Targets. At the end of an exchange of messages, both the Initiator and Target agree to maintain, for retransmitting only, a minimum number of Information Packets (greater than or equal to one).

In the absence of an explicit maximum Information Packet length agreement for a value larger than 256, the implied maximum Information Packet length agreement shall be 256 for both Initiators and Targets. At the end of an exchange of messages, both the Initiator and Target agree to maintain Information Packets of at least the negotiated size. The minimum value for this field shall be 256.

Table 16 - PACKET TRANSFER REQUEST message format

Byte	Value	Description
0	01h	Extended Message Format Code
1	0Ah	Extended Message Length
2	07h	Extended Message Code
3 - 4	0000h	Reserved
5	xxh	Originator's Packet Offset Count
6 - 7	xxxxh	(MSB) Originator's Maximum Information Packet Length (LSB)
8	xxxxh	Reserved
9	xxh	Responder's Packet Offset Count
10 - 11	xxxxh	(MSB) Responder's Maximum Information Packet Length (LSB)

The originator's packet offset count represents n+1 Information Packets that the originator requests be retained by the responding GPP device. A maximum of 256 Information Packets may be retained per task at the end of an PTR message negotiation (packet offset count value of 255). For Initiators which implement the MODIFY DATA POSITION message, there may be an additional requirement to retain additional Information Packets to support that message..

The originator's maximum Information Packet length plus one that the originator requests be transmitted by the responding GPP device.

The responder's packet offset count represents n+1 Information Packets that the responder requests be retained by the originating GPP device. A maximum of 256 Information Packets may be retained per task at the end of an PTR message negotiation. This field shall be set to zero in the first message sent by an originator in a PTR message exchange. The responder's maximum Information Packet length plus one that the responder requests be transmitted by the originating GPP device. This field shall be set to zero in the first message sent by an originator in a PTR message exchange.

The packet transfer agreement becomes invalid after any condition which may leave the packet transfer agreement in an indeterminate state such as:

- after a reset event
- after a TARGET RESET message and
- after a power cycle.

In addition, an GPP device may initiate an PTR message exchange whenever it is appropriate to negotiate a new packet transfer agreement.

The PTR message exchange establishes the minimum number of Information Packets and the maximum length of each Information Packet that the originating GPP device and the responding GPP device shall independently maintain to

permit retransmitting Information Packets in case of physical or logical errors in Information Packets. (See the RESEND PREVIOUS Information Packet message.)

The originator (the GPP device that sends the first of the pair of PTR messages) sets its values according to its capabilities to retain Information Packets on a per task basis. The originator shall set the responder's values to zero. If the responding GPP device can retain Information Packets at the originator's requested level, the responder returns the same values in its PTR message in the originator's fields. If it requires a smaller packet offset count or a shorter maximum Information Packet length, it substitutes a lower value in the appropriate originator field(s) of its PTR message.

Independently, the responder to the original PTR message fills in the responder's packet offset value and the responder's maximum Information Packet length value.

The originator analyzes the response and may transmit another PTR message with the same or lower originator's values. The originator analyzes the responder's packet offset count value and the responder's maximum Information Packet length value. If the originating device can also retain Information Packets at the responder's level, it returns the same values in its PTR message in the responder's fields. If it requires a smaller retention count or a shorter Information Packet maximum length, it substitutes an lower value in the appropriate responder's field(s) of its PTR message. Successful negotiation occurs when both devices exchange the same packet offset value and maximum Information Packet length value or the packet offset value reaches 0 and the maximum Information Packet length reaches 256, whichever occurs first, in both sets of fields (i.e., identical values in respective fields).

Each device shall retain at least the number of Information Packets of less than or equal to the maximum Information Packet length for retransmission, as agreed upon in the most recent PTR message exchange. If there is any error in the negotiation process, the packet offset count agreement shall be 1 and the maximum Information Packet length shall be 256 after the task for both pairs of values.

7.2.12 REPORT PATH STATUS

The REPORT PATH STATUS message is a singular message. The REPORT PATH STATUS message (table 17) requests or reports the status of the path on which the connect was made relative to the path group naming level and status of a path group. The response message provides to the requestor the status of the path and the path group.

Path status may be requested by any Initiator of any Logical Unit. Any Target acting as an Initiator may request path status of the Logical Unit in an Initiator used for the processor device class support.

Table 17 - REPORT PATH STATUS message format

Byte	Value	Description
0	01h	Extended Message Format Code
1	10h	Extended Message Length
2	0Ch	Extended Message Code
3 - 4	00h or 08h	WWNi Length
4 - 11	xxh	(MSB) WWNi (LSB)
12	xxh	ImplPath
13	xxh	PathOthr
14	xxh	Ungrpd
15	xxh	Grouped
16 - 17	0000h	Reserved

When this message is used to request path status, the following fields shall be set to zero: WWNi length, ImplPath, PathOthr, Ungrpd, and Grouped. The WWNi field shall be ignored. All reserved fields shall also be set to zero. All remaining fields are set as specified in table 17.

When this message is used as a response to a path status request, the fields in the message contain values as specified below.

If the WWNi Length field is set to 8, the WWNi field contains a valid WWNi for the path. A value of zero for the WWNi field means that the WWNi field shall be ignored. When the PathOthr, Ungrpd, or Grouped field is set to one, the WWNi length field shall be set to 8 and the WWNi field shall contain a valid value. If the ImplPath bit is set to one, the WWNi length field shall be set to 0 in the response message.

When the WWNi length field is set to 8, the WWNi field contains the WWNi value transferred by a SET WWN message that was successfully processed.

The state of the path may be any one of the following:

1 If the implicitly named path (ImplPath) bit is set to one, no WWNi has been accepted from the Initiator to any Logical Unit in this Target using this path. A task received on a path in this state shall operate in single path mode.

If an WWNi has been established for the path this field shall be set to zero.

2 If the Path to Other LUs (PathOthr) bit is set to one, an explicitly named path to at least one Logical Unit on this path, other than the one identified for the nexus, exists. This is functionally equivalent to the report above, but it provides additional information to the Initiator. A task received on a path in this state shall operate in single path mode.

If no explicitly named path exists for a different Logical Unit, this field shall be set to zero.

3 If the Ungrouped (Ungrpd) bit is set to one, an explicitly named path to the selected Logical Unit exists, but it is not currently part of an established path group. A task received on a path in this state operates in single path mode. If the path is grouped, this field shall be set to zero.

4 If the Grouped bit is set to one, an explicitly named path to the selected Logical Unit exists and is currently in the grouped state. The path group can consist of one or more paths. A task received on this path may respond on any path in this path group unless single path status is in effect for the path group or multiple path operation has been temporarily suspended for the task. If the path is ungrouped, this field shall be set to zero.

Based on the definitions of the fields given above, the ImplPath, PathOthr, Ungrpd, and Grouped fields are mutually exclusive. Only one field shall be set to one in the response message.

7.2.14 RESEND PREVIOUS INFORMATION PACKET

The RESEND PREVIOUS INFORMATION PACKET message is a singular message. The RESEND PREVIOUS INFORMATION PACKET message (table 18) is sent by an Initiator or a Logical Unit to indicate that one or more previously transferred Information Packets must be resent. The packet transfer request negotiation has established an agreement for the maximum number of packets to be retained by both the Initiator and the Logical Unit.

Table 18 - RESEND PREVIOUS INFORMATION PACKET message format

Byte	Value	Description
0	01h	Extended Message Format Code
1	06h	Extended Message Length
2	05h	Extended Message Code
3	00h	Reserved
4	xxh	Previous Packet Number
5 - 7	000000h	Reserved

A value of zero in the previous packet number field causes the device to resend the last Information Packet transferred. Values of 1 through 255 represent the 2nd to 256th most recent packets transmitted where retransmission is to start. The logical element receiving the RESEND PREVIOUS INFORMATION PACKET message shall resend all Information Packets, in order, starting with the indicated Information Packet.

It shall be an error for a GPP device to transmit a value in the previous packet number field which is logically larger than the limit negotiated using the PACKET TRANSFER REQUEST message. If this error is detected, the logical element processing this message shall send an Information Packet to the other logical element which contains an EXTENDED MESSAGE REJECT message pointing to the Previous Packet Number field in error.

The Information Packet containing an RESEND PREVIOUS INFORMATION PACKET message shall contain only one ILE, a message ILE.

7.2.15 SET WWN

The SET WWN message is a singular message. The SET WWN message (table 19) has six functions:

1 to explicitly name the two logical elements using the path on which a connect occurred;

2 to add a path to an established path group;

3 to remove a path from an established path group;

4 to establish a path group;

5 to disband a path group;

6 to send a response to a SET WWN request to confirm the action taken by the receiver.

A path shall be explicitly named before it can be included in an established path group. Naming the paths in a path group does not establish the path group. If an established path group is required, it shall be explicitly established using this message with the EstabPG option correctly specified.

Establishing a path group is the mechanism for enabling multiple path operations in logical system and allowing assignment for multiple path groups. Including a path in a path group does not restrict access to the logical unit from any other path.

Table 19 - SET WWN message format

Byte	Value	Description
0	**01h**	**Extended Message Format Code**
1	**12h**	**Extended Message Length**
2	**0Dh**	**Extended Message Code**
3	**00h or 08h**	**WWN Length**
4 - 11	**xxh**	**(MSB) WWN (LSB)**
12	**xxh**	**NamePath**
13	**xxh**	**AddPath**
14	**xxh**	**RmvPath**
15	**xxh**	**EstabPG**
16	**xxh**	**SnglStus**
17	**xxh**	**DsbndPth**
18	**xxh**	**Reserved**
19	**xxh**	**Request Status**

If the WWN length field is set to 8, the WWN field contains a valid WWN for the logical element sending the message. A value of zero for the WWN length field means that the WWN field shall be ignored. See the request status field below for reporting invalid values.

When the WWN length field is set to 8, the WWN field contains the WWN to be established for the path. The WWN is not an attribute of the GPP port used to send the SET WWN message; it is an attribute of the logical element controlling the port.

NOTE — For GPP devices that implement ports using SPI, the WWN value may be a manufacturer code concatenated with the serial number of the logical element. For GPP devices that implement ports using Fibre Channel, the WWN can be the Node_Name of the controlling Node behind the N_Port of the logical element. If a GPP device implements both types of ports, the Fibre Channel Node_Name is unique and is the guaranteed unique to be placed in the WWN field.

That is, all GPP ports in the same logical element use the same WWN in this message so that grouping of several paths is possible. This also permits multiple path operations.

The only valid values for the NamePath, AddPath, RmvPath, EstabPG, SnglStus, DsbndPth fields are zero and one. See the request status field below for reporting invalid values.

When the name path (NamePath) field is set to one, the message is used to name the path on which he connect is made. The WWN length field shall be set to the value 8. See the request status field below for reporting invalid values. The WWN field contains the WWN of the logical element. The interface control prefix fields in the Information Packet

contain the remaining information to completely name the path to a Logical Unit. The LUN in the Information Packet must be valid for the Target. The path remains in an ungrouped state.

Explicitly naming a path to a Logical Unit does not alter access privileges (i.e., assignment). The AddPath field may be set to zero or one. If the NamePath field is set to one, and the RmvePath, the EstabPG, or DsbndPth field is set to one, see the request status field below for reporting invalid combinations of values. The SnglStus field shall be ignored.

When the name path (NamePath) field is set to one and the path already has been given a different name, see the request status field below for reporting an attempt to change the WWN. When the name path (NamePath) field is set to zero, the path name function shall not be invoked in the Logical Unit.

When the add path (AddPath) field is set to one, the named path which has been previously named or is named with this message when the NamePath field is set to one shall be added to an established path group of the same name. A path may be added to a path group after the path group has been established by sending this message from the Initiator on the path to be added to the path group. The attributes of the path group shall be transferred to the new path. The path enters the grouped state. Adding a path to a path group causes the assignment status of the path group to be transferred to the new path. The NamePath field may be zero or one. When the NamePath field is set to one, the function shall be executed before the AddPath function. If the AddPath field is set to one and the RmvePath field, the EstabPG field, or the DsbndPth field is set to one, see the request status field below for reporting invalid combinations of values. The SnglPath field shall be ignored. If the WWN does not match an established path group, see the request status field below for reporting an attempt to add a path group that does not currently exist. If the path does not have a name assigned and the AddPath field is set to one, see the request status field below for reporting an attempt to add a path group that does not currently exist.

A path may be removed from an established path group by a message from the Initiator on the path to be removed from the path group. When the remove path (RmvePath) field is set to one, the named path which has been previously named, shall be deleted from the established path group of the same name. The attributes of the path group shall be removed from the path. The path enters the ungrouped state. The path retains the WWN provided through the add path function. Removing a path from a path group removes assignment for that path if assignment exists for the path group.

A path group is implicitly disbanded when the last path is removed from an established path group using the RmvePath function. The logical path remains as it did before the path group was established. If the RmvePath field is set to one and the NamePath field, AddPath field, the EstabPG field, or the DsbndPth field is set to one, see the request status field below for reporting invalid combinations of values. The SnglPath field shall be ignored. If the WWN does not match an established path group, see the request status field below for reporting an attempt to remove a path from a path group that does not currently exist. If the path does not have an WWN set and the RmvePath field is set to one, see the request status field below for reporting an attempt to add a path group that does not currently exist.

A set of named paths, a logical path, shall be established as a path group when the establish path (EstabPG) field is set to one. The value of the single path status (SnglStus) field is used in conjunction with the EstabPth field as specified below. The status reporting attribute can only be changed by disbanding the path group and reestablishing it with the desired attribute. Each path in the newly established path group enters the grouped state.

Establishing a path group for a Logical Unit does not alter access privileges. The NamePath field may be zero or one. When the NamePath field is set to one, the function shall be executed before the EstabPG function. If the EstabPG field is set to one and the AddPath field, the RmvePath field, or the DsbndPth field is set to one, see the request status field below for reporting invalid combinations of values. If the WWN does not match the name of one or more paths in an ungrouped state, see the request status field below for reporting an attempt to establish a path group that does not currently exist.

If the EstabPth field is set to one and the single path status (SnglStus) field is set to one, the path group is established with the attribute that any ACA shall be reported only on the path in the established path group where the initial connection was made for each task. This is called single path status mode. Any report status may be returned on any valid path in the established path group.

If the EstabPth field is set to one and the single path status (SnglStus) field is set to zero, the path group is established with the attribute that any ACA may be reported on any path in the established path group. This is called multiple path status mode. Any report status may be returned on any valid path in the established path group.

The inverse of establishing a path group is to disband a path group. A path group shall be explicitly disbanded when the disband path group (DsbndPth) field is set to one and the message is received from the Initiator on any path in the established path group. Each path in the path group enters the ungrouped state. Each path retains the WWN provided through the add path function. Disbanding a path group removes assignment for each path if assignment exists for the path group. If the DsbndPth field is set to one and the NamePath field, AddPath field, the RmvePath field, or the EstabPG field is set to one, see the request status field below for reporting invalid combinations of values. The SnglPath field shall be ignored. If the WWN does not match the name of an established path group, see the request status field below for reporting an attempt to disband a path group that does not currently exist.

When the SET WWN message is a request, the request status field shall be set to zero. See the request status field below for reporting invalid values in as SET WWN response message.

When the SET WWN message is a response, the request status field shall have one of the values specified below:

1 A value of 00h indicates that the request was processed correctly by the Logical Unit.

2 A value of 01h indicates an invalid combination one of the fields NamePath, AddPath, RmvPath, EstabPG, DsbndPth was detected in the message.

3 A value of 02h indicates an in invalid value in the Password Length field.

4 A Value of 03h indicates an invalid value in the NamePath, AddPath, RmvPath, EstabPG, SnglStus, or DsbndPth fields.

5 A value of 04h indicates that the request status field was not zero in the request message.

6 A value or 05h indicates that a request to add a path, remove a path, establish a path group or disband a path group was received on a path that does not currently have a valid WWN. The task terminates.

7 A value of 06h indicates that an attempt was made to set a second WWN of a different value. The new WWN is not accepted. The task terminates.

8 Values 07h through FFh are reserved.

7.2.16 TARGET RESET

The TARGET RESET message is a singular message. The TARGET RESET message is sent from an Initiator to direct a Target to clear all tasks on that GPP device. This message shall not be sent from a Target to a Initiator. This message forces a reset condition in the selected GPP device. The Target shall create a unit attention condition for all Initiators for each logical unit.

The Information Packet containing a TARGET RESET message shall contain only one ILE, a message ILE.

All pending status and data for that logical unit for all Initiators shall be cleared. No status or message shall be sent for any of the tasks. A unit attention condition shall be generated for all Initiators. When reporting the unit attention condition the additional sense code shall be set to POWER ON OR RESET OR TARGET RESET. Previously established conditions, including MODE SELECT parameters, assignments, CONTROL ACCESS passwords, and ACA shall be reset by the TARGET RESET message.

7.2.17 TASK COMPLETE

The TASK COMPLETE message is sent from a Logical Unit to an Initiator to indicate that the execution of an task has completed and that valid status has been sent to the Initiator. This message shall not be sent from a Target to a Initiator. The task may have completed successfully or unsuccessfully as indicated by the status value.

The TASK COMPLETE message when present in an Information Packet, shall be the last, or only, ILE in the Information Packet.

7.2.18 TASK WAITING

The TASK WAITING message is a singular message. The TASK WAITING message (table 20) is sent to indicate that an active task requires additional Information Packets. The message is used to direct an Initiator or a Logical Unit to continue sending Information Packets for the identified task. This message does not make a specific request for any type of ILE.

The message contains a count of the packet credit count available to the other logical element. The packet credit count is valid until a packet is received. The Initiator or Logical Unit has processed all transmitted Information Packets and is waiting for more Information Packets to be transmitted (i.e., an underrun condition).

Table 20 - TASK WAITING message format

Byte	Value	Description
0	01h	Extended Message Format Code
1	06h	Extended Message Length
2	09h	Extended Message Code
3	00h	Reserved
4 - 5	xxxxh	(MSB) Sequence Number of the Last Information Packet Processed (LSB)
6 - 7	xxxxh	Packet Credit Available

The Sequence Number of the Last Information Packet Processed field indicates the last Information Packet sequence number processed. The task cannot continue until additional Information Packets are transferred.

The Packet Credit Available field contains the count of the available number of Information Packet slots available to hold Information Packet transmitted by the other logical element. The count is a general count and does not apply to any specific task, but rather the count represents free preallocated space to receive Information Packets for any task.

The Information Packet containing a TASK WAITING message shall contain only one ILE, a message ILE.

In response, the Initiator or Logical Unit picks the path it wants to use based on agreements with the Initiator for the task and sends the additional Information Packets.

7.2.19 TERMINATE TASK

The TERMINATE TASK message is a singular message. The TERMINATE TASK message (table 21) is sent from the Initiator to the Logical Unit to advise the Logical Unit to terminate the task without corrupting the logical data of the logical unit. This message shall not be sent from a Target to a Initiator. Upon successful receipt of this message the Logical Unit shall terminate the identified task as soon as possible and return TASK TERMINATED status. The sense key shall be set to NO SENSE and the additional sense code and the additional sense code qualifier shall be set to TASK TERMINATED. The TERMINATE TASK message shall not affect pending status, data, and commands for other tasks. However, continued execution and status of other tasks for the H_C_x_y nexus may be affected by the task set error recovery option specified in the control mode page parameters.

Table 21 - TERMINATE TASK message format

Byte	Value	Description
0	01h	Extended Message Format Code
1	03h	Extended Message Length
2	08h	Extended Message Code
3 - 5	000000h	Reserved
6 - 7	xxxxh	(MSB) Last Information Packet to Process (LSB)

The last Information Packet to process field indicates the last Information Packet that the Initiator requires that Logical Unit process for the terminated task. The Logical Unit shall clear all Information Packets with a larger value that the value of this field. If processing has progressed beyond this value, the Logical Unit shall immediately terminate all processing for the task and enter the ACA state for the logical unit.

This message is normally used by the Initiator to stop a lengthy read, write, or verify operation when a higher priority task is available to be executed.

The Information Packet containing an TERMINATE TASK message shall contain only one ILE, a message ILE.

If the task that is being terminated has a logical data transfer associated with it, the valid bit in the sense data shall be set to one and the information field shall be set as follows:

1 If the command descriptor block specifies an allocation length or parameter list length in bytes, the information field shall be set to the difference (residue) between the transfer length and the number of bytes successfully transferred.

2 If the command descriptor block specifies a transfer length field, the information field shall be set as defined in the REQUEST SENSE command.

If the task being terminated has no data transfer associated with it, the Logical Unit shall set the valid bit in the sense data to zero and terminate the task with TASK TERMINATED status. The sense key shall be set to NO SENSE and the additional sense code and the additional sense code qualifier shall be set to TASK TERMINATED.

When any error condition is detected while terminating a task the Logical Unit shall ignore the TERMINATE TASK message and terminate the task with the appropriate error status and sense data for the error condition.

If the Logical Unit completes all processing for an task (i.e., all data has been read, written, or processed) and a TERMINATE TASK message is received before the task is terminated, the Logical Unit shall ignore the TERMINATE TASK message and terminate the task in the normal manner.

If the Logical Unit receives a TERMINATE TASK message before any command descriptor block is transferred or the Information Packet identifies an H_C_x_y nexus that does not exist at that time, the Logical Unit shall set the valid bit in the sense data to zero and terminate the task with TASK TERMINATED status. The sense key shall be set to NO SENSE and the additional sense code and the additional sense code qualifier shall be set to TASK TERMINATED.

If the affected task is in the task set (an H_C_L Task for untagged or an H_C_L_Q Task for tagged) and has not started execution, the Logical Unit shall record the event with the task and wait until the task starts executing and then terminate the task. The Logical Unit shall terminate the task with TASK TERMINATED status. The sense key shall be set to NO SENSE and the additional sense code and additional sense code qualifier shall be set to TASK TERMINATED. The valid bit shall be set to zero.

The TERMINATE TASK message provides a means for the Initiator to request the Logical Unit to reduce the transfer length of the current command to an amount that has already been transferred. The Initiator can use the sense data to determine the actual number of bytes or blocks that have been transferred.

7.2.20 UNASSIGN

The UNASSIGN message (table 22) is used to remove assignment for a path or path group from the set of assigned paths or path groups and the message is used to release assignments for extents and elements. The message is also used as a response to the requestor of the action taken by the receiver.

Table 22 - UNASSIGN message format

Byte	Value	Description
0	**01h**	**Extended Message Format Code**
1	**0Eh**	**Extended Message Length**
2	**0Eh**	**Extended Message Code**
3	**xxh**	**WWNi Length**
4 - 11	**xxh**	**(MSB) WWNi (LSB)**
12	**xxh**	**Unassignment Type**
13	**xxh**	**Unassignment Identification**
14	**00h**	**Reserved**
15	**xxh**	**Request Status**

Access privileges may be unassigned by this message when it is received for any path group to which assignment currently exists on any path in any established path group which also has assignment.
If the UNASSIGN message is received on a path for which assignment does not exist, the message shall be rejected by returning a response message with the request status field set to FDh. All other fields are set to the same value as in the request message.

The two functions of the UNASSIGN message are:

1 When the UNASSIGN message is used to request unassignment, the WWNi length field shall have either a value of zero or 8. If the WWNi length field contains any other value the response message shall be returned with an request status field value of FEh. If the WWNi length field value is zero, the Logical Unit shall unassign the identified Logical Unit, extents or elements, for the established path group on which the message was received. See the unassignment type field.

2 If the WWNi length is set to 8 and the WWNi field matches the name of another established path group which has assignment, the Logical Unit shall remove assignment of the Logical Unit, extents or elements, for the other established path group. See the unassignment type field. If the WWNi field does not match the WWNi name of another established path group for the Logical Unit, the response message shall be returned with an request status field value of FCh.

When the UNASSIGN message is used to report the result of an unassignment request, all fields shall be the same value as in the original request if the unassignment is made. If the unassignment is not made for any reason other than as specified above with specific request status values, the value in the request status field shall be FFh to indicate that the unassignment request failed.

The WWNi field consists of an 8 byte hexadecimal value which uniquely names the path group for which assignment is to be made. The contents of the field are valid only when the WWNi Length field is set to 8.

The unassignment type field values identify the type of unassignment requested:

- 00h Logical Unit unassignment

- 01h extent unassignment

- 02h element unassignment

- 03h-FFh reserved

If the unassignment type field is any reserved value, the message shall not be accepted and the request status field of the response message shall be set to FBh.

The unassignment identification is meaningful only when the unassignment type field is other than 00h. When the unassignment type field value is 00h, this field shall be ignored. When the unassignment type is other than 00h and the unassignment is successful with a match on the unassignment identification field value, the Logical Unit shall remove the assignment identification field value and the assignment. When an UNASSIGN message is processed the unassignment identification value identifies which assignment is to be terminated.

The request status field shall be 00h on each assignment request. The request status field shall be 00h in a response message when the assignment is successful. Other valid values in this field in a response message are specified in this subclause.

8 GPP services for a logical element

GPP services assume that an Information Packet is the unit of transfer between two GPP devices. All physical interface considerations are hidden from the Initiators and Logical Units. GPP services process requests for transfer or receipt of various data types between two GPP devices and manage their transmission between these devices.

8.1 GPP services introduction

GPP services specify protocols for each side of GPP as follows:

- GPP device to GPP services
- GPP services to the service delivery interface.

Figure 10 shows the relationship of these two protocols to the components of the SAM standard. GPP services reside in each GPP device and they are responsible for the transfer of information between two GPP devices. The GPP device is isolated from its service delivery interface by GPP services.

8.1.1 Logical element-to-GPP services introduction

A GPP device need not be aware of the service delivery subsystem (SDS) used to transfer Information Packets. Each Information Packet may be transferred on a different SDS. For GPP, these protocols involve specifying the various data types transferred and presenting each data type to the destination GPP device as specified in clauses 4 through 7.

- GPP services form Information Packets at the source GPP device and decompose Information Packets at the destination GPP device at the request of the corresponding GPP devices.

- The service delivery interface performs all operations related to a physical interface and the protocols associated with that interface.

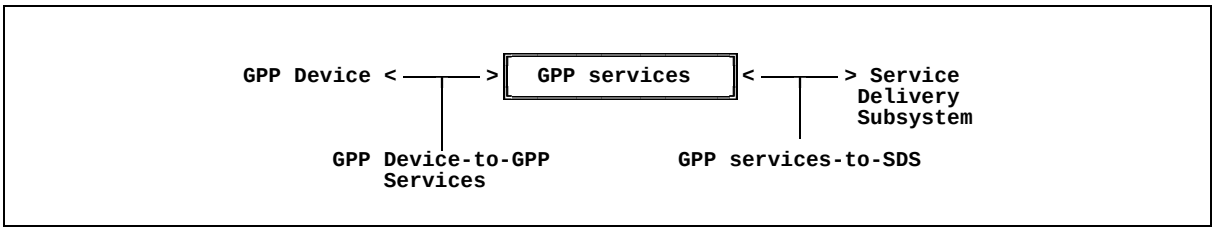


Figure 10 - GPP Information Packet transfer services

8.1.2 GPP services-to-SDS introduction

The GPP services-to-SDS protocols are tailored to the specific implementation of a physical interface at each GPP port. This interface isolates each GPP device from most, if not all, SDS specific knowledge and accommodation. Operational control of the GPP port for GPP operations is assigned to GPP services. The major clauses identified below specify:

- GPP services-to-Fibre Channel services (see annexes A and B)
- GPP services-to-SPI Services (see annexes C and D)
- GPP services-to-Internet Services (see annexes E and F)
- GPP services-to-HIC Services (see annexes G and H)
- GPP services-to-ATM AAL 5 Services (see annexes I and J)

In some devices, multiple logical protocols may be used through the same port (e.g., a host using GPP and the Internet Protocol (IP) through Fibre Channel). If this is the case, operational control of the port is multiplexed between the various independent services with GPP services managing only GPP transfers.

8.2 GPP-device-to-GPP services

These services are used by a GPP device to cause an Information Packet to be constructed and transferred to another GPP device. This service interface is patterned after the Common Access Method (CAM) standard of SCSI-2 and the GPP services-to-Fibre Channel service interface in annex A.

The intended level of this interface is similar to the CAM SCSI Interface Module which is capable of processing a linked list of SCSI I/O REQUEST CAM Control Blocks (CCBs). The structure of a CAM CCB is altered to be symmetrical for the Initiator and Target modes required in each GPP device.

The following service functions support GPP Information Packet transfers:

- GPP_TASK.request
- GPP_TASK_TAG.indication
- GPP_TASK.indication
- GPP_TASK.confirmation

An example using these services to manage a GPP Task is shown in figure 11. This example results in Information Packets flowing in both directions. This example could represent several read-type commands sent to a disk device and the corresponding logical data, status, and completion messages returned by the disk device.

A GPP_TASK.request specifies that one or more Information Packets associated with a Task are to be processed or that a Task Management function is to be performed. One GPP_TASK.request may result in several Information Packets being exchanged between GPP devices.

The GPP_TASK_TAG.indication is made available to track requests and to be able to identify a Task to GPP services. If some Task Management function is to be performed for that Task while being processed by GPP services, the GPP device uses the value returned in the indication to identify the Task in GPP services.

GPP_TASK.confirmation may indicate successful completion of the service request or that an error not recoverable by GPP services has occurred.

GPP_TASK.indication is the response given to the destination GPP device that Information Packets have arrived. Partial Information Packets or Information Packets received by GPP services in error are not given an indication at the destination GPP device.

Each component of the GPP device-to-GPP services interface is specified below.

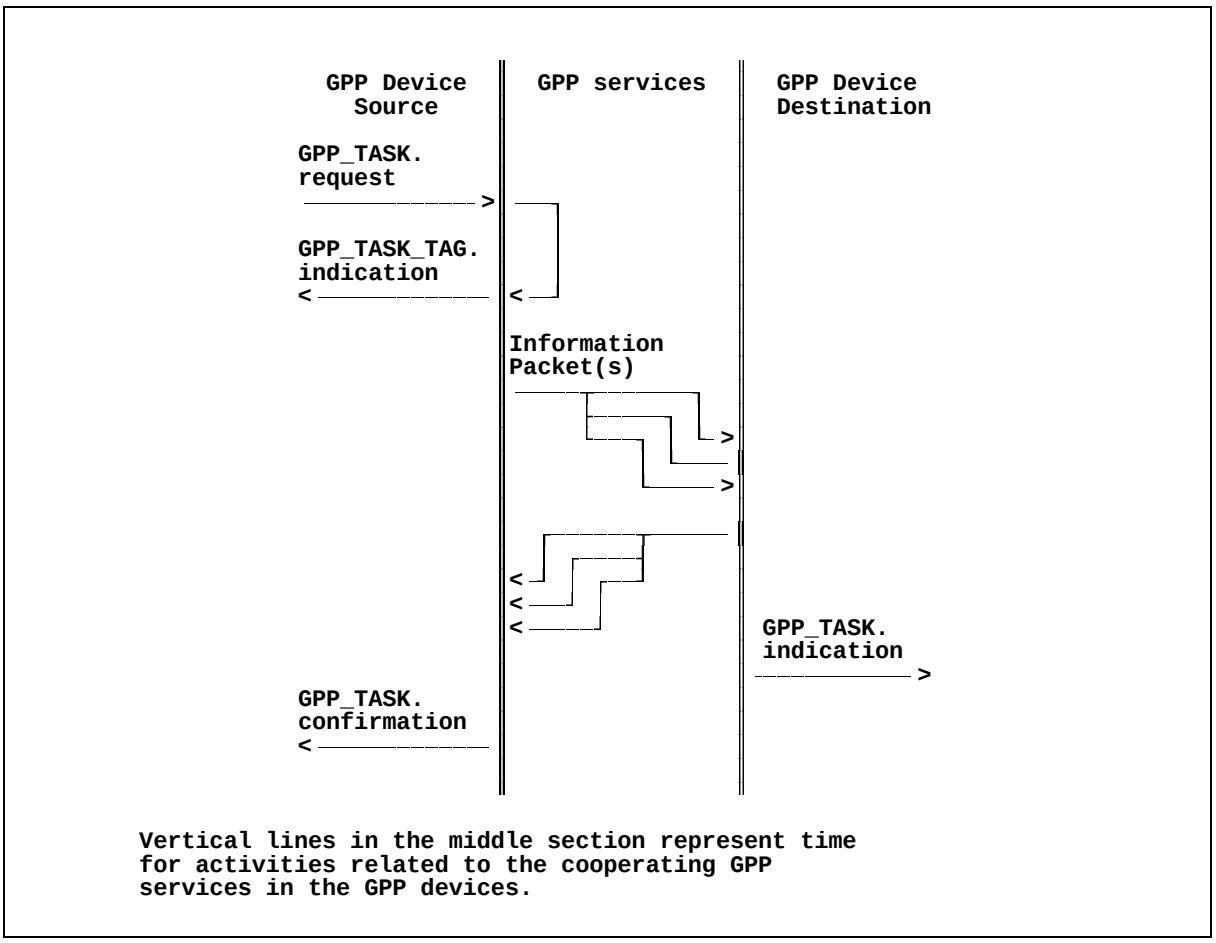


Figure 11 - Information Packet transfer functions example

8.2.1 GPP_TASK.request

The GPP_TASK.request function (figure 12) specifies sufficient information for managing one or more information transfers between a local GPP logical element to a single peer GPP logical element (i.e., no broadcast or multicast functions are supported).

8.2.1.1 Semantics

The abbreviation TRB (for Task request block) is used to shorten the parameter names in figure 12. A TRB specifies one or more actions to be performed by GPP services. Two or more TRBs may be linked together by pointers to form one large request.

```
GPP_TASK.request (

     /* Fixed Length Portion of Request */

     Address_of_this_TRB,
     Next_TRB_Pointer,
     Function_Code,
     Task_Tag,
     TRB_Status,
     TRB_Flags,
     Peripheral_Driver_Pointer,
     Request_Mapping_Info,
     Callback_on_Completion,
     Timeout_Value,
     Vendor_Unique_Flags,
     Private_Data,
     Task_Element_List_Pointer )


Task_Element_List (

     /* Fixed Length Portion of Task_Element_List */

     S_GPP_ID,
     D_GPP_ID,
     LU_Handle,
     Tagged_Group,
     Task_Type,
     Direct_Transfer,
     Use_Multipath,
     Allow_SPVR_Fcns,
     Number_of_Element_Descriptors,
     Element_Descriptor_Pointer)      /* Only or 1st in linked list */


Element_Descriptor (

     Element_Descriptor_Pointer,      /* NULL for only/last element in list        */
     Transfer_Function,
     Function_Specific_Parms,
     Element_Status,
     Number_of_SG_List_Entries,
     Element_Transfer_Length,
     Element_Residual_Length,
     BP_SG_List_Pointer )             /* Only or 1st in linked list */


SG_List (

     BP_SG_List_Pointer,              /* NULL for only/last element in list        */
     SG_Transfer_Length,
     SG_Residual_Length,
     Buffer_Pointer )
```

Figure 12 - GPP_TASK.request semantics

Figure 13 shows the structure and relationship of the various lists associated with a GPP_TASK.request. A request may be a single TRB or several TRBs may be linked together to form a complex request. The set of TRBs forms one GPP_TASK.request.

The various components of a GPP_TASK.request are composed of linked lists of lists. This permits flexible arrangement of the various ILEs in the request and avoids fixed position control structures that might hamper transporting GPP services to multiple environments. GPP does not specify the field sizes, but rather it indicates the order and relationship of the various fields. Each Element_Descriptor is permitted to include a variable length scatter-gather list to identify the source or destination of bytes for that ILE.

The list structure permits flexible placement of information by GPP devices and may eliminate the need for multiple moves of the bytes within a GPP device. The TRB structure permits information to be taken from or written to storage locations directly from GPP services. The structure allows GPP services to point to the ILEs from their position in a received Information Packet without further movement until needed by the GPP device.

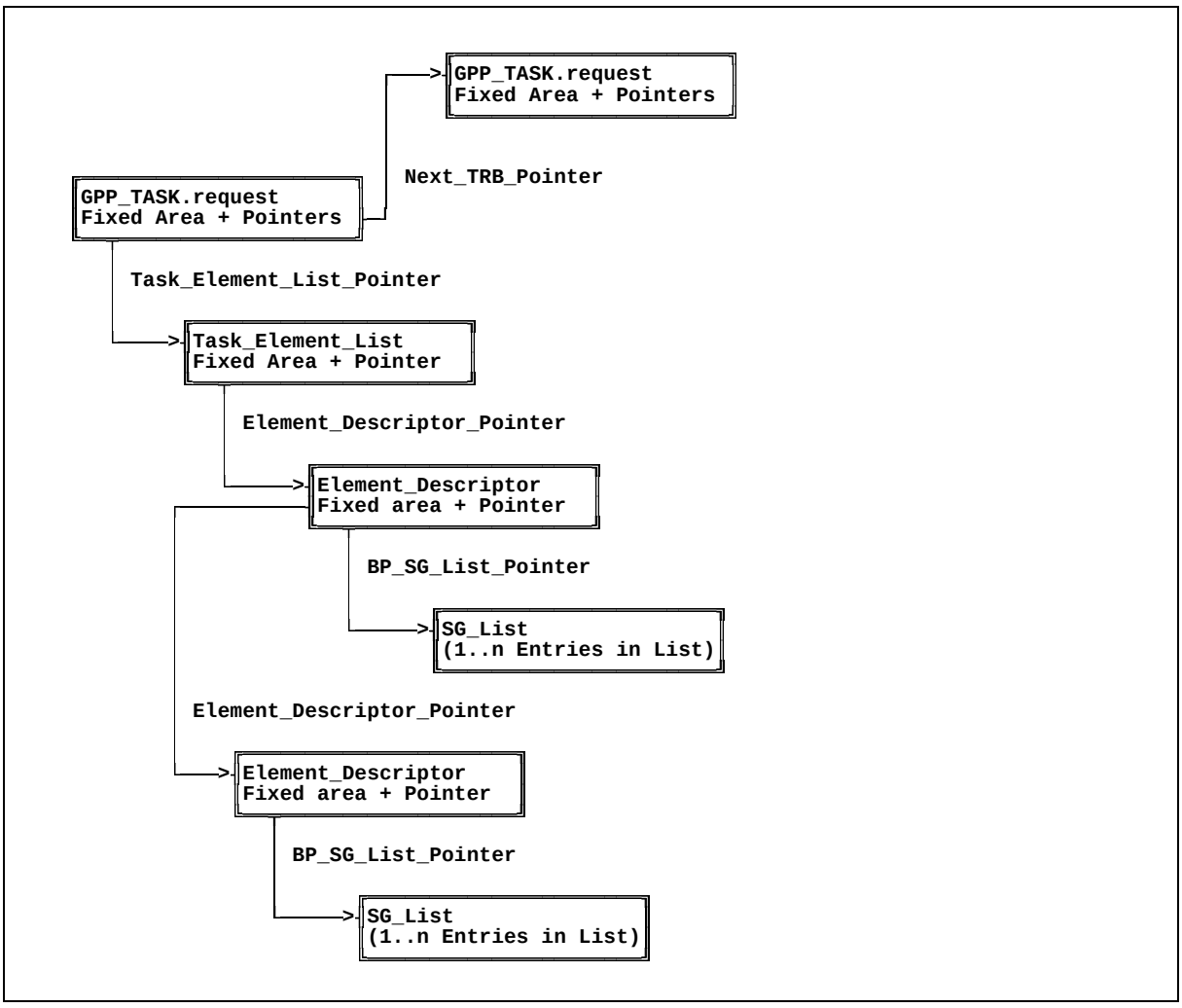


Figure 13 - GPP_TASK.request pointers

8.2.1.2 When generated

This function is invoked by a GPP device to request a Task Management function, a GPP Task, or portion of a GPP Task be performed with another GPP device. Each GPP_TASK.request may be for a different Task. A single GPP_TASK.request shall be for a single Task or Task Management function. If linked commands are used, each command may be managed by using one GPP_TASK_.request per command. Alternatively, several commands may be processed in one GPP_TASK.request.

8.2.1.3 Effect of receipt

Upon receipt of a GPP_TASK.request function, the source GPP services perform the following actions:

1 Indicate to the GPP device a unique Task tag (Task_Tag) using the GPP_TASK_TAG.indication function. If this is a continuation of an earlier Task, the indication shall return the same Task_Tag value. The GPP_TASK_TAG.indication may be implemented by placing a non-zero value in the Task_Tag field. This permits the GPP device to read this field without requiring a callback operation.

2 Validate the parameters and verify that the requested operation is possible. Set the TRB_Status to REQUEST IN PROGRESS if verification is successful.

3 Build one or more Information Packets to support the transfer of information to the destination GPP services. The requirements for forming an Information Packet and splitting the data associated with an Element_Descriptor into multiple ILEs is contained in clauses 5 through 7.

4 Request a transfer of the Information Packets by the service delivery interface. Place these Information Packets on a sent queue and maintain the depth of this queue based on the current agreement with the other GPP device. The initial agreement is one Information Packet of 256 bytes.

5 Track the progress of the Task. If GPP services receives any Information Packets from the destination GPP services for this Task, the local GPP services moves data to the indicated locations or points to the received data as specified in the Element_Descriptors.

6 Use the GPP_TASK.confirmation function to notify the GPP device that the request has been successfully completed or abnormally terminated. See the TRB_Status parameter field.

8.2.1.4 GPP_TASK.request parameter usage

The parameters in the TRB section of the request are roughly equivalent to the CAM transport layer operating system dependent areas of a SCSI-2 CAM CCB. They have been rearranged to collect the fields in one place for easy management and change across environments. The last parameter is a pointer to the GPP specific information of the request.

The Address_of_this_TRB parameter provides a confirmation to the GPP Service that it is operating on the correct TRB. If the pointer to the TRB and this field value differ, an error is reported and the TRB is not processed. All activity in one TRB is for exactly one Task or Task Management function.

The Next_TRB_Pointer parameter, when not null, indicates that at least one additional TRB follows this TRB. The next TRB may not be for the same Task. If this pointer is not null, GPP services performs the same test for the Address_of_this_TRB as specified above. When all TRB pointers and Address_of_this_TRB fields have been checked and found to be consistent, the request may be processed further.

The Function_Code parameter indicates the type of operation to be performed. The Function_Code actions are roughly equivalent to CAM function code actions, but the required functions have been expanded to include all SAM Task Management functions and all optional function codes are specified as implementation dependent:

- Execute Initiator I/O (Initiator mode)

- Execute Target I/O (Target mode)

- Task Management (see 8.2.1.6 for specific functions)

- Other Function_Code values, modeled after CAM, may be used to invoke other CAM transport layer functions in GPP services, but they are beyond the scope of GPP services.

The Task_Tag parameter is set by GPP services to a unique value that may be used by the GPP device for tracking purposes.

- For Task Management functions that affect a single Task identified by a previously uncompleted GPP_TASK.request, this value shall be the same value.

- For a new GPP_TASK.request, not associated with an uncompleted GPP_TASK.request, this field shall contain zero.

- For a GPP_TASK.request that is continuing a Task in progress, the value in this field shall be the same as that reported in the GPP_TASK.indication for that Task.

The TRB_Status parameter indicates the status of the entire TRB. The initial value supplied by the GPP device is REQUEST PENDING. When GPP services begins to process the TRB, the TRB_Status is changed to REQUEST IN PROGRESS. Other values indicate completion with or without errors and are not specified by GPP services (see SCSI-2 CAM Status for sample values).

The TRB_Flags parameter contains a set of fields to indicate special handling of the TRB. The values are not specified by GPP services (see SCSI-2 CAM Flags for sample values).

The Peripheral_Driver_Pointer parameter contains a value supplied by the Peripheral Driver in SCSI-2 CAM and shall not used by and shall not be changed by GPP services.

The Request_Mapping_Info parameter (operating system dependent (OSD)) contains a value that shall not be used by and shall not be changed by the GPP services.

The Callback_on_Completion parameter (OSD) contains the method by which GPP services is to return to the GPP device when the TRB activity is complete (successful or abnormal completion).

The Timeout_Value parameter contains the maximum time value in seconds that this TRB may remain uncompleted. A value of FF....FFh in this parameter indicates there is no maximum time that this request can remain uncompleted. The elapsed time shall be measured from the point that the REQUEST IN PROGRESS value is placed into the TRB_Status field.

The Vendor_Unique_Flags parameter provides for vendor specific information to be provided to GPP services. The format and semantics of this field are beyond the scope of GPP services.

The Private_Data parameter provides an area for GPP services to use when performing its operations. This area may be used to report SG_List entries for receive-type Element_Descriptors when no SG_List is supplied.

The Task_Element_List_Pointer parameter provides a pointer to the Task_Element_List for this TRB. This field shall not be null since the Task_Element_List is a required parameter for each TRB.

8.2.1.5 Task_Element_List parameter usage

The S_GPP_ID parameter indicates the GPP device that is the source of the request. This value is used for the first part of the identification of a Task or Task Management function.

The D_GPP_ID parameter indicates the GPP device of the destination of the TRB. This value is used for the second part of the identification of a Task or Task Management function.

The LU_Handle parameter provides an interface independent value that identifies the Logical Unit in the Target associated with this TRB. This value is used for the third part of the identification of a Task or Task Management function.

The Tagged_Group parameter indicates that a tagged Task type should be used if supported by the Logical Unit. If not, GPP services shall emulate the equivalent behavior to the best of its ability. See the Task_Type parameter, below. This value is used to select the value for the Tag field that makes up the fourth part of the identification of a Task or Task Management function. GPP services shall assign the Tag value, for all Tasks. Valid values are:

- YES
- NO

The Task_Type parameter identifies the type of Task management behavior requested of the Logical Unit if the Tagged_Group field value is YES. The valid values for this parameter are:

- HEAD OF QUEUE
- ORDERED
- SIMPLE
- ACA

The Direct_Transfer parameter requests GPP services to use its fastest means of transfer for this Task. This request is advisory for GPP services and may have no effect on the operation.

NOTE — Some SDSs have multiple data rates that are selectable by the service delivery interface. This parameter indicates only that, if available, the maximum available rate should be used. This is an advisory field.

The Use_Multipath parameter indicates the level of multipathing that is allowed for this Task. This is a request to GPP services. If multiple paths are set up between the Initiator and Target, the Initiator may request that this function be used. If the request is to not use the feature, GPP services shall not use multiple paths for the operation. If the request is to use multiple paths, GPP services may use multiple paths for the operation if multiple paths are available and active between the two GPP devices. Requesting multiple path operation is advisory for GPP services. The resulting operation may or may not used multiple paths. The valid values are:

- YES
- NO

The Allow_SPVR_Fcns parameter identifies the types of commands and Task management functions that are allowed in the Task. If this parameter is set to NO, the Logical Unit does not execute any command or function identified as supervisory. This parameter is advisory to GPP services. If set to NO, GPP services complies. If the parameter is set to YES, GPP services may override the request if the source of the request is not authorized to make such a request. The implementation of this control is Initiator specific and is not specified by GPP services. GPP makes no check of the CDB operation codes for compliance with this parameter. The valid values are:

- YES
- NO

NOTE — This parameter provides a system management tool to prevent certain operations by unauthorized components of a system. This parameter is primarily aimed at the multiple path set up and management as well as the controlled access functions.

The Number_of_Element_Descriptors parameter provides a count of the number of Element_Descriptor_Pointers that are pointed to by the Element_Descriptor_Pointer field. This value shall be at least one.

The Element_Descriptor_Pointer points to the first Element_Descriptor related to this Task. The Element_Descriptors are formed in a linked list.

- Send-type Element_Descriptors are transmitted in order.

- Receive-type Element_Descriptors are processed in order as Information Packets are received for the Task.

- A special exception occurs when a Recv_AUTOSENSE and the command status does not require a status transfer. In this case, GPP services marks the Element_Status as unused and continues.

- All send-type Element_Descriptors may be processed at one time and transferred to the other GPP device. GPP services tracks the progress of the Task through receive-type Element_Descriptors.

8.2.1.6 Element_Descriptor parameter usage

Both the Initiator and the Logical Unit attempt to predict the correct order of transfer of ILEs for each command. The Element_Descriptor_Pointer points to the next Element_Descriptor or is null for the last descriptor.

Completion messages (TASK COMPLETE, LINKED COMMAND COMPLETE, and LINKED COMMAND COMPLETE (with flag)) are detected by GPP services and not presented to the GPP device. The effect of these messages is to continue or terminate a Task. The final status is reflected in the Task_Tag field.

The Transfer_Function identifies the specific transfer to be made or received at this point in the Task. The transfer function values are specified below. The value in parentheses after each entry name is S or R denoting send or receive, respectively, and the GPP ILE type value that corresponds to that transfer function value.

Send_MGMT_FCN (S0) This Transfer_Function value indicates that a Task Management function is requested. In GPP, all Task Management functions are implemented as message ILEs.

The valid values for the Function_Specific_Parms parameter are:

- TASK WAITING
- ABORT TASK
- ABORT TASK SET
- CLEAR ACA
- CLEAR TASK SET
- TARGET RESET
- TERMINATE TASK
- All other messages

Send_CDB (S1) This Transfer_Function value indicates that a Command Descriptor Block (CDB) is to be transferred.

The valid values for the Function_Specific_Parms parameter are:

- Data_Direction values are:
  - NO CPD, NO CRD, NO LD;
  - CPD OR LD(out) FOLLOWS CDB;
  - CRD OR LD(in) FOLLOWS CDB.

Send_CPD (S2) This Transfer_Function value indicates that command parameter data is associated with the CDB. A Send_CPD Element_Descriptor shall be the next Element_Descriptor following a

Send_CDB Element_Descriptor when the CDB has parameter data to be transferred to the Target.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Send_CRD (S3) This Transfer_Function value indicates that command response data is to be transferred to the Initiator. A Send_CRD Element_Descriptor shall be the next Element_Descriptor following a Send_CDB Element_Descriptor when the CDB has response data to be transferred to the Initiator.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Send_LD (S4) This Transfer_Function value indicates that logical data is to be transferred. A Send_LD Element_Descriptor shall be the next Element_Descriptor following a Send_CDB Element_Descriptor when the CDB has logical data to be transferred for the command.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Send_STATUS (S5) This Transfer_Function value indicates that status is to be transferred to the Initiator.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Send_AUTOSENSE (S6) This Transfer_Function value indicates that autosense data is to be transferred to the Initiator and is related to the immediately preceding status transfer for the same Task.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Recv_MGMT_FCN (R0) This Transfer_Function value indicates that the Target expects to receive a Task Management function. In GPP, Task Management functions are implemented as message ILEs. Other messages are received using this same Transfer_Function value.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Recv_CDB (R1) This Transfer_Function value indicates that the Target expects to receive a CDB.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Recv_CPD (R2) This Transfer_Function value indicates that the Target expects to receive command parameter data.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Recv_CRD (R3) This Transfer_Function value indicates that the Initiator expects to receive command response data.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Recv_LD (R4) This Transfer_Function value indicates that either the Initiator or Target expects to receive logical data.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Recv_STATUS (R5) This Transfer_Function value indicates that the Initiator expects to receive status.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

Recv_AUTOSENSE (R6) This Transfer_Function value indicates that the Initiator expects to receive autosense. If the command being executed does not terminate with a condition that results in an autosense transfer, this Element_Descriptor Element_Residual_Length is set equal to the Element_Transfer_Length parameter value (i.e., no bytes transferred). An indication that no bytes were transferred does not mean that there is no sense data for the status transferred in the immediately preceding status Element_Descriptor. The Target may be waiting for an explicit request from the Initiator to transfer the sense data for some condition.

The Function_Specific_Parms parameter is reserved for this Transfer_Function value and shall be set to zero.

The Function_Specific_Parms parameter provides specific additional information for each Transfer_Function. See Transfer_Function definitions above for specific sub fields and values.

The Element_Status parameter provides a means for GPP services to indicate the status of each Element_Descriptor in a TRB. This permits the GPP device to identify the exact status of a complete Task by examining the individual Element_Status values. If the order of request receipt does not match the anticipated order of Element_Descriptor receipt, the GPP_TASK.request is terminated but the Task remains active in GPP services. The GPP device is notified of the discrepancy in the Transfer_Function type in the Element_Status where the difference was detected. The GPP device may supply an alternate Element_Descriptor list in a new GPP_TASK.request for the same Task_Tag value.

The Number_of_SG_List_Entries parameter identifies the number of entries in the SG_List. This value is greater than zero on send-type Element_Descriptors when a scatter/gather list is present and may be zero for receive-type Element_Descriptors. If the parameter value is zero for a send-type Element_Descriptor, The Element_Transfer_Length, Element_Residual_Length and BP_SG_List_Pointer are redefined to mean the same as a SG_List SG_Transfer_Length, SG_Residual_Length, and Buffer_Pointer, respectively.

The Element_Transfer_Length specifies the maximum transfer length for receive-type Element_Descriptors. For send-type Element_Descriptors, this is the actual number of bytes to be transferred. If a scatter/gather list is supplied, the GPP device must supply a total transfer length in the SG_List entries equal to or greater than the Element_Transfer_Length value.

The Element_Residual_Length parameter specifies the residual transfer length for this Element_Descriptor. The value may be any value from zero to the Element_Transfer_Length value. The parameter shall be set to zero by the GPP device in each request. GPP services places the residual value for each Element_Descriptor in this field to be returned to the requesting GPP device.

The BP_SG_List_Pointer contains:

- a null value (receive-type Element_Descriptors only)

- a Buffer_Pointer value for the data (when not using an SG_List)

- a pointer to a scatter/gather list.

The number of entries in the scatter/gather list is indicated by the Number_of_SG_List_Entries parameter value.

- The individual items in the list are linked together in a linked list.

- The value is null when the request is to reside in the Information Packet assembly area for receive-type transfer functions. GPP services places a pointer here to its list of SG_LIST entries accumulated from one or more Information Packets.

The examples below provide templates that may be used for all SCSI commands and Task Management functions from the Initiator point of view. Each single command or Task Management request fits one of these models. Multiple Element_Descriptors are indicated for data types that may be broken into bursts. In this case, each Element_Descriptor would identify the source or destination of data for each burst.

A TEST UNIT READY command in a GPP_TASK.request consists of the following Element_Descriptors:

- Send_CDB
- Recv_STATUS
- Recv_AUTOSENSE (1 or more)

A READ-type command in a GPP_TASK.request consists of the following Element_Descriptors:

- Send_CDB
- Recv_LD (1 or more)
- Recv_STATUS
- Recv_AUTOSENSE (1 or more)

A WRITE-type command in a GPP_TASK.request consists of the following Element_Descriptors:

- Send_CDB
- Send_LD (1 or more)
- Recv_STATUS
- Recv_AUTOSENSE (1 or more)

A MODE SELECT command in a GPP_TASK.request consists of the following Element_Descriptors:

- Send_CDB
- Send_CPD (1 or more)
- Recv_STATUS
- Recv_AUTOSENSE (1 or more)

A REQUEST SENSE command in a GPP_TASK.request consists of the following Element_Descriptors:

- Send_CDB

- Recv_CRD (1 or more)
- Recv_STATUS
- Recv_AUTOSENSE (1 or more)

An ABORT TASK SET Task Management function in a GPP_TASK.request consists of the following Element_Descriptor:

- Send_MGMT_FCN

8.2.1.7 Scatter/Gather List (SG_LIST) parameter usage

The SG_List consists of a one set of three fields that may be linked to additional entries. Each set shall contain all four fields. The SG_List represents a gather list for send-type Transfer_Function parameter values. The SG_List represents a scatter list for receive-type Transfer_Function parameter values. This symmetrical structure gives each GPP device control over the source of or placement of bytes for each Element_Descriptor.

The BP_SG_List_Pointer is either a null value for the last entry in the list or is a valid pointer to the next entry in the list. Scatter/gather list entries shall be processed in the order supplied by the GPP device.

The SG_Transfer_Length parameter indicates the maximum number of bytes that may be taken from or placed in the buffer pointed to by the Buffer_Pointer parameter. If a scatter/gather list is supplied, the GPP device must supply a total transfer length in the SG_List entries equal to or greater than the corresponding Element_Transfer_Length.

The SG_Residual_Length parameter contains the residual length in bytes of any unused portion of this SG_List entry.

The Buffer_Pointer parameter contains a pointer to the start of the bytes to be transferred or received.

NOTE — Alignment of the start of buffers and the length of buffers is an implementation issue. Since many ILEs do not correspond to system word lengths, all systems must provide a means for single byte access for their storage environment.

8.2.2 GPP_TASK_TAG.indication

This function provides an indication of the Task_Tag used for identifying a GPP_TASK.request.

8.2.2.1 Semantics

Figure 14 shows the semantics of this indication.

```
GPP_TASK_TAG.indication (
                        Task_Tag )
```

Figure 14 - GPP_TASK_TAG.indication semantics

8.2.2.2 When generated

This function is generated in response to the GPP_TASK.request function for a new request. Continuation of a Task with an additional GPP_TASK.request shall not produce a new indication.

8.2.2.3 Effect of receipt

The GPP Device relates the Task_Tag value to the corresponding GPP_TASK.request. The Task_Tag value is reported again on the GPP_TASK.confirmation to relate the response to the original GPP device GPP_TASK.request.

8.2.2.4 GPP services managed parameters

The Task_Tag provides the local identifier of the GPP_TASK.request being processed by GPP services.
NOTE — An implementation of this function may be to place the value in the Task_Tag field of the original GPP_TASK.request. The value is then available to the GPP device through a reference to this field in its original request.

8.2.3 GPP_TASK.indication

This function indicates the receipt of an Information Packet for which no matching GPP_TASK.request was found at the destination GPP services. A match is based on fields in the Information Packet header
that identifies the Task or Task Management function and the Information Packet sequence number within the Task.

If an available GPP_TASK.request is found, it shall be used and a GPP_TASK.confirmation shall be reported instead of the GPP_TASK.indication.

NOTE — This structure permits anticipatory operation on the part of both the Initiator and the Target. If operations are correctly anticipated, a performance enhancement may result. If there is no anticipation or a mismatch in anticipated activity, the same performance results.

8.2.3.1 Semantics

Figure 14 shows the semantics of this indication.

```
GPP_TASK.indication (
                    S_GPP_ID,
                    D_GPP_ID,
                    LU_Handle,
                    Tagged_Request,
                    Task_Type,
                    Direct_Transfer,
                    Use_Multipath,
                    Allow_SPVR_Fcns,
                    Number_of_Element_Descriptors,
                    Task_Valid )
```

Figure 15 - GPP_TASK.indication semantics

8.2.3.2 When generated

This function is generated after an Information Packet is successfully received. Information Packet are reported in the order successfully received which is not necessarily the order of transmission by the other GPP device. GPP provides internal reordering of Information Packets within a Task.

GPP services perform the following actions:

1 Reports information from the valid Information Packet header to the GPP device so that the GPP device can respond with a GPP_TASK.request to process the contents of the Information Packet. No validation of the Information Packet contents is made other than to determine that the internal structure is correct and counting the minimum number of Element_Descriptors required to process the Information Packet.

2 Uses the GPP_TASK.indication function to inform the GPP device that an Information Packet is waiting in GPP services.

8.2.3.4 Effect of receipt

The GPP device informs the correct Logical Unit or the Target. A default, or stub, function must be provided for all Logical Units.

The GPP device processes the GPP_TASK.indication by sending a GPP_TASK.request to support the minimal functions assigned to a Target. This includes, but is not limited to responding to INQUIRY and REQUEST SENSE commands.

8.2.3.5 GPP services managed parameters

The S_GPP_ID parameter indicates the GPP device that is the source of the request. This parameter is used for the first part of the Task identification.

The D_GPP_ID parameter indicates the GPP device of the desired destination of the Task. This parameter is used for the second part of the Task identification.

The LU_Handle parameter provides an interface independent value that identifies the Logical Unit in the Target involved in the Task. This parameter is used for the third part of the Task identification.

The Tagged_Request parameter indicates that a queued Task type should be used if supported by the Logical Unit. See the Task_Type parameter, below. This value is used for the fourth part of the Task identification. Valid values are:

- YES
- NO

The Direct_Transfer parameter reflects the type of request observed at the local GPP services. Some service delivery interfaces can detect the type of request. High performance requests, as determined by GPP services are reflected in this parameter. Valid values are:

- YES
- NO

The Task_Valid parameter indicates that GPP services identified the Information Packet with a valid Task, but that there were no receive-type TRBs to use. Valid values are:

- YES
- NO

8.2.3.6 GPP_TASK.indication semantics

The Task_Type parameter identifies the type of Task Management behavior requested of the Logical Unit. This parameter is used for the fourth part of the Task identification. The valid values for this parameter are:

- HEAD OF QUEUE
- ORDERED
- SIMPLE
- ACA

The Use_Multipath parameter indicates the level of multipathing that is allowed for this Task. This is an advisory report from GPP services. If multiple paths are set up between the Initiator and Target, the Initiator may request that multiple paths be used or not. If the request is to not use the feature, GPP services does not use multiple paths. If the request is to use multiple paths, GPP services may use the feature if multiple paths are active between the two GPP devices. The valid values are:

- YES
- NO

The Allow_SPVR_Fcns parameter identifies the types of commands and Task Management functions that are allowed in a Task. If this parameter is set to NO, the Logical Unit shall not execute any command or function identified as supervisory. If this parameter is set to YES, supervisory type commands are permitted within the Task. The valid values are:

- YES
- NO

The Number_of_Element_Descriptors parameter provides a count of the minimum number of Element_Descriptor_Pointers needed to process the Information Packet. This value shall be at least one.

The examples below provide templates that may be used for all SCSI commands and Task Management functions from the Logical Unit point of view. Each example shows one or more GPP_TASK.requests per command. Each SCSI command type or Task Management function fits one of these templates. Multiple Element_Descriptors are indicated for data types that may be broken into bursts. In this case, each Element_Descriptor would identify the source or destination of data for each burst.

Since all GPP Tasks begin with either a CDB or a Task Management function, the requirement to match the elements received to the Element_Descriptors in a GPP_TASK.request in order is given an exception for matching the first and second Element_Descriptors of a GPP_TASK.request. If a Task Management function is first, the Recv_CDB Element_Descriptor is marked as not used. If a CDB is first, the Recv_MGMT_FCN Element_Descriptor is marked as not used.

A TEST UNIT READY command in a GPP_TASK.request consists of the following Element_Descriptors:

First GPP_TASK.request

- Recv_MGMT_FCN
- Recv_CDB

Second GPP_TASK.request

- Send_STATUS
- Send_AUTOSENSE (0 or more)
- Send_MGMT_FCN (completion message)

A READ-type command in a GPP_TASK.request consists of the following Element_Descriptors:

First GPP_TASK.request

- Recv_MGMT_FCN
- Recv_CDB

Second GPP_TASK.request

- Send_LD (1 or more) - may be zero for errors
- Send_STATUS
- Send_AUTOSENSE (0 or more)
- Send_MGMT_FCN (completion message)

A WRITE-type command in a GPP_TASK.request consists of the following Element_Descriptors:

First GPP_TASK.request

- Recv_MGMT_FCN
- Recv_CDB
- Recv_LD (1 or more)

Second GPP_TASK.request

- Send_STATUS
- Send_AUTOSENSE (0 or more)
- Send_MGMT_FCN (completion message)

A MODE SELECT command in a GPP_TASK.request consists of the following Element_Descriptors:

First GPP_TASK.request

- Recv_MGMT_FCN
- Recv_CDB
- Recv_CPD (1 or more)

Second GPP_TASK.request

- Send_STATUS
- Send_AUTOSENSE (0 or more)
- Send_MGMT_FCN (completion message)

A REQUEST SENSE command in a GPP_TASK.request consists of the following Element_Descriptors:

First GPP_TASK.request

- Recv_MGMT_FCN

- Recv_CDB

Second GPP_TASK.request

- Send_CRD (1 or more) - may be zero for errors
- Send_STATUS
- Send_AUTOSENSE (0 or more)
- Send_MGMT_FCN (completion message)

A TASK WAITING Task Management function in a GPP_TASK.request consists of the following Element_Descriptor:

- Send_MGMT_FCN

8.2.4 GPP_TASK.confirmation

This function provides an appropriate response to a GPP_TASK.request indicating the success or failure of the request.

8.2.4.1 Semantics

Figure 16 shows the semantics of the confirmation.

```
GPP_TASK.confirmation (
                    Task_Tag,
                    Last_Element_Descriptor_Processed_Pointer,
                    Transmission_Status,
                    Reject_Reason )
```

Figure 16 - GPP_TASK.confirmation semantics

8.2.4.2 When generated

This function is generated upon the completion of an attempt to process a GPP_TASK.request. A confirmation may also be sent for uncompleted requests related to Task Management functions.

8.2.4.3 Effect of receipt

The logical element may continue with the Task or begin a new Task.

8.2.4.4 GPP services managed parameters

The Task_Tag parameter provides the local identifier of the Task in GPP services.

The Last_Element_Descriptor_Processed_Pointer parameter identifies, by pointer value, the last Element_Descriptor processed for the Task identified by the Task_Tag value.

The Transmission_Status parameter provides a service response for the request that indicates request status as one of the following:

- Successful - request processed completely; the type of command completion is indicated in an associated status ILE. Task completion, if applicable is indicated in the TRB_Status field of the TRB.

- Task Management action, Service delivery failure or recipient failure

    - Unsuccessful - request was not processed completely due to a service delivery failure or Task Management action.

    - Stopped_by_Recipient - Recipient stopped Information Packet transfer.

    - Rejected_Request - The request was not completed due to the specified Reject_Reason.

When the Transmission_Status parameter value is other than Successful, the Reject_Reason parameter indicates one of the following reason codes.

- Destination Reject

- Physical Transmission Reject

- Destination Busy

- Physical Transport Busy

- Task Management action

- GPP Services detected content error

- Others - not specified by GPP.

8.3 GPP services-to-SDS

The annexes that follow provide the mapping to cause Information Packets to be transported across the identified SDSs. GPP manages the activities of a service delivery interface for GPP activities only. Where a service delivery interface supports multiple protocols, it is shared by the various protocol services, like GPP services.

Each SDS is describes in two annexes.

- The first annex gives the mapping rules to cause Information Packet to be transported across the SDS. This is a normative annex.

- The second annex provides usage guidelines about a particular SDS related to performance, overall task management, etc. This is an informative annex.

ANNEX A
(normative)

FC-PH usage for GPP

This annex specifies the characteristics of Fibre Channel useful to the SCSI-3 Generic Packetized Protocol (GPP). Fibre Channel specifies the physical requirements for interconnection. For specific details on the Fibre Channel, see the Fibre Channel standards. See Clause 2 for the complete titles and numbers for these standards.

This annex and annex B specify the use of Fibre Channel as a transport layer for GPP. A logical system may be constructed using any of the service delivery subsystems in GPP or a combination of two or more.

A.1 Fibre Channel definitions for GPP

A.1.1 class of service: A general definition of the methods by which communication circuits are maintained between communicating N_Ports. The classes of service supported by an N_Port or the Fabric are one determining factor in establishing inter-operability between Ports.

A.1.1 Class 1 service: A dedicated connection service. Class 1 service does not reflect the peer-to-peer nature of SCSI since the connection between two N_Ports is held until released. That is, the disconnect privilege is not granted in a logical sense to a Target. With a Fabric, the switch connections are set and maintained, whether used or not, until explicitly released. Frames received in a certain order at the receiver had been transmitted in the same order by the transmitter. Other attributes of Class 1 service are: required Login, an establish/break connection protocol is required, end-to-end confirmation of frame transmission, maximum frame size limitations, dual flow control, and busy or reject link continue and link responses are not allowed once the connection is made.

A.1.1 Class 2 service: A frame multiplex service. Class 2 service most nearly reflects the peer-to-peer nature of SCSI. Frames transmitted in a certain order at a transmitter may not arrive in the same order at the receiver. Other attributes of Class 2 service are: required Login, connectionless service, frame multiplexing at the sender or receiver, end-to-end confirmation of frame transmission, maximum frame size limitations, dual flow control, and busy and/or reject link continue and link responses are allowed.

A.1.2 CRC field: A four-byte field at the end of the frame content field. The CRC field contains information which may help determine if transmission errors occur across a link, a Fabric, or passing through any N_Port or F_Port. This field is normally generated by a hardware element and one or more hardware elements check it at the receivers. The CRC field value is calculated before frame transmission begins. The algorithm for generating the field is specified in the appropriate standard.

A.1.3 D_ID: See Destination Identifier.

A.1.4 Destination ID: See Destination Identifier.

A.1.5 Destination_Identifier: An identifier unique to one N_Port on a GPP interface. The identifier is placed in the frame header field of each frame of an Information Unit to identify the destination N_Port for an Information Unit.

A.1.6 Exchange: A set of one or more Information Units defined to perform some action for GPP. Information unit content and the ordering of Information Units is a function of GPP. An Exchange operates between the Originator and Responder N_Ports where the connect was made. Exchanges operate in a unidirectional mode for GPP.

A.1.7 Exchange protocol: A set of one or more Exchanges defined for GPP to carry out its functions. GPP provides rules to translate the activities for each nexus into a set of one or more Information Units. The Exchange protocol is independent of the logical content of the Information Packets (Information Units). That is, two or more logical

functions may use the same Exchange protocol. A set of Exchange protocols is called an FC-4 mapping in the Fibre Channel.

A.1.8 F_Port: A component of a Fabric element which connects to the end of a link. An F_Port receives frames from an N_Port and directs them into the Fabric. An F_Port receives frames from the Fabric and transmits them to an N_Port.

A.1.9 Fabric: A Fibre Channel entity with at least one Fabric element. A Fibre Channel implementation does not require a Fabric.

A.1.10 Fabric element: A component of a Fabric which, at minimum, performs frame switching or circuit switching. Some Fabric elements may perform both functions and may perform the N_Port naming function called Fabric Login. Fabric elements attach to each other via links to provide more sophisticated interconnect subsystem than a single Fabric element is designed to handle.

A.1.11 Fabric class of service: The class(es) of service supported by a Fabric.

A.1.12 fibre: A single transmission element which transfers frames in a unidirectional manner.

A.1.13 frame: The smallest unit of information transmitted across a link. A frame consists of Idle Words, a start of frame delimiter, Frame Content field, an end of frame delimiter, and Idle Words. A frame, excluding Idle Words, is from 36 to 2148 characters long. An N_Port sends frames in the order established by the Exchange protocol selected. Frames may not arrive at a receiving N_Port in the order transmitted, depending on the makeup of the Fabric, if present, and the class of service.

A.1.14 Frame Content field: A set of three fields in a frame: the frame header field, frame data field, and CRC field. The Frame Content field varies from 28 to 2140 characters in multiples of four characters.

A.1.15 frame data field: A variable length field, always a multiple of 4 bytes, which conveys logical information across a link. A frame data field has a zero minimum length and a maximum length of 2112 bytes. The frame header field has a field designated to identify the number of pad bytes, if any, in a frame data field. The concatenation of frame data fields extracted from each Information Unit contains an Information Packet.

A.1.16 frame header field: A fixed length and fixed format structure, positioned after the start of frame delimiter, used by the Fabric and N-ports during transmission of frames. The frame header field provides physical addresses for frame routing on the Fibre Channel. The frame header field has other fields which help determine protocol errors.

A.1.17 Idle Word: An ordered set chosen to represent the idle state of a fibre (i.e., no frame being transmitted). A primitive Sequence may substitute for an Idle Word in some instances.

A.1.18 link: From the point of view of an N_Port, a link is a physical interface with two fibres and the associated connectors at each end.

A.1.19 Login: A procedure for each N_Port to identify itself to, or gain its identity from, the Fabric or another N_Port. An N_Port performs a Login operation with each N_Port with which it intends to communicate and successfully exchange service parameters. Each N_Port Exchanges service parameters with its F_Port if the Fabric is present.

A.1.20 Logout: A procedure for removing the service parameter agreement between two N_Ports. When the procedure is complete, Login must occur before an N_Port can successfully perform GPP operations.

A.1.21 N_Port: A port on the Fibre Channel which attaches to a link end point. Each N_Port attaches, through a link, to exactly one F_Port, if the Fabric exists. An N-port is equivalent to a GPP port. In Fibre Channel, the same N_Port may be used for other protocols as well.

A.1.22 N_Port class(es) of service: The class(es) of service supported by an N-port.

A.1.23 N_Port service parameters: A set of parameters specifying the capabilities of an N_Port. N_Ports exchange service parameters with an N_Port or F_Port before attempting to use GPP Exchanges.

A.1.24 Node: A facility composed of one or more N_Ports.

A.1.25 Node_Name: An identifier which uniquely identifies a Node on in a GPP logical system. See WWNi and WWNt in 3.1.

A.1.26 Originator: The N_Port responsible for initiating an Exchange. When two N_Ports are communicating either may become an Originator and start an Exchange. The Originator of an Exchange may be in either Initiator role or Target role.

A.1.27 Originator Exchange status: A set of fields and values which permits an N_Port to monitor an Exchange for proper operation (e.g., a violation of frame ordering). The Originator maintains this status for each active Exchange.

A.1.26 OX_ID: See X_ID.

A.1.27 point-to-point topology: A Fibre Channel implementation where each N_Port can communicate with exactly one other N_Port using one link.

A.1.28 point-to-point operation: A function of the Fibre Channel which logically connects two N_Ports, when a Fabric is interposed, to make them appear connected as in a point-to-point topology.

A.1.29 Responder: The N_Port that supports an Exchange initiated by an Originator. When two N_Ports are communicating, either may become a Responder to an Exchange initiated by the other. The Responder of an Exchange may be in either Initiator role or Target role.

A.1.30 RX_ID: See X_ID.

A.1.31 S_ID: See Source_Identifier.

A.1.32 Sequence: A protocol in the Fibre Channel specifying how to transmit a set of frames between N_Ports. In GPP, a Sequence transmits a single Information Unit which carries one Information Packet. Fibre Channel specifies rules for determining when each Sequence ends. Fibre Channel specifies rules whereby the receiver can reassemble an Information Packet in the correct order.

A.1.33 service parameters: A set of field values specifying the capabilities of an N_Port or F_Port. Communicating ports exchange service parameters before attempting GPP Exchanges.

A.1.34 Source_Identifier: An identifier unique to an N_Port on a GPP interface placed in the frame header field of each frame of an Exchange to identify the sender of the frame.

A.1.35 X_ID: Exchange_Identifier. The Exchange_Identifier for an Exchange. "X_ID" may be prefixed by the letters "O" or "R" meaning "Originator X_ID" and "Responder X_ID," respectively.

A.2 Fibre Channel specific messages

There are no Fibre Channel specific messages defined for GPP.

A.3 Fibre Channel physical description for GPP

GPP devices attached to Fibre Channel as a service delivery interface using a 2-fibre connection called a link. The Fibre Channel standard defines implementations using different types of conductors. All Fibre Channel operations are conducted through primitive signals and primitive Sequences sent on the fibres or through the transfer of Information Units.

A service delivery interface in each GPP device is called an N_Port. A GPP device may have more than one N_Port. The physical port in a Fabric element is called an F_Port. In a Fabric topology, each N_Port is connected by a single link to exactly one F_Port.

Fibre Channel supports the arbitrated loop topology that is also compatible with GPP. The arbitrated loop topology forms a loop within which devices communicate by sharing the bandwidth of the loop. No fabric element is needed with the arbitrated loop topology. Fibre Channel ports for an arbitrated loop are called NL_Ports.

GPP devices may connect to:

- a point-to-point topology (see FC-PH).

- arbitrated loop topology (see FC-AL).

  A GPP device capable of being used in an arbitrated loop topology is also be capable of operating in the point-to-point and the switched point-to-point topologies. The arbitrated loop topology requires extensions to the basic N_Port specified in FC-PH.

- a switched point-to-point topology (see FC-SW).

  Fabric elements may exist which permit inter-operability between different fibre types and different transmission rates. Each N_Port attaches directly to the fabric and the fabric routes frames between N_Ports.

Each GPP device shall support Fibre Channel classes of service as follows (See FC-PH, Annex S, S.1.1):

- Class 2 service

- Class 1 service

Support for Class 3 service is outside the scope of GPP.

In SCSI, all information flow is given a direction from the point of view of an Initiator. The flow of information in SCSI is outbound from Initiators and inbound from Targets. Because of the full-duplex nature of Fibre Channel operations, the actual difference in the direction of information flow can only be determined by examining the Information Packet content. The physical transport layer is symmetrical in all transfer operations. Figure 17 shows the basic items of Fibre Channel which affect GPP operations.

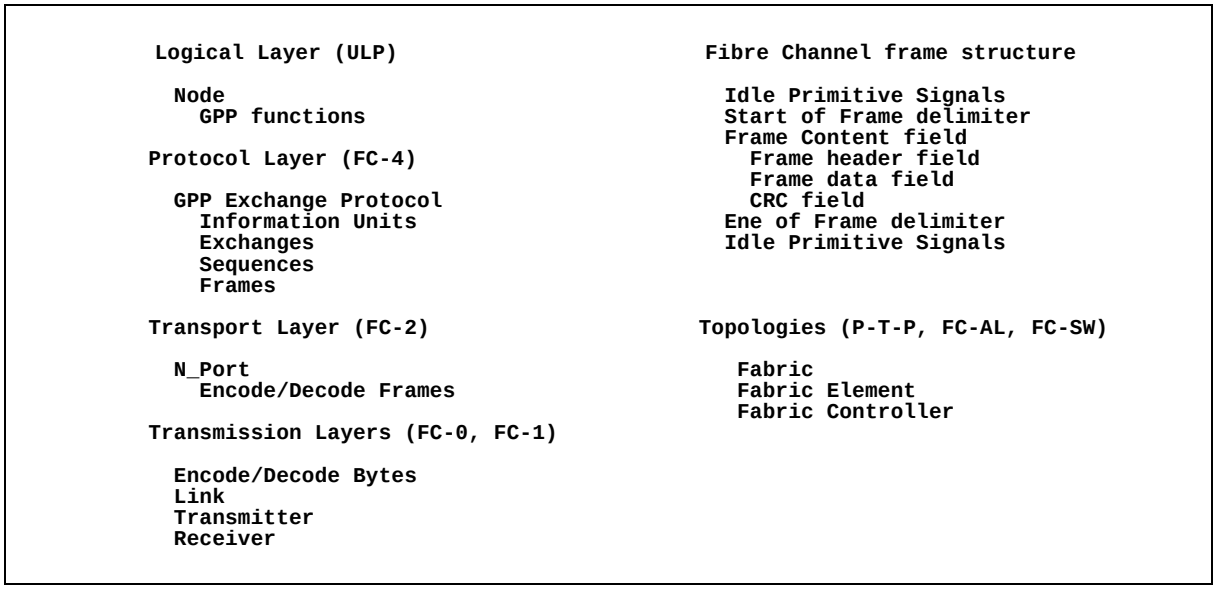


Figure 17 - Basic Fibre channel terms

A.4 FC-PH encapsulation of an Information Packet

The unit of work between an Initiator or a Target in GPP is an Information Packet. The equivalent FC-PH construct is called a Information Unit. The FC-PH Information Unit delivery protocol forms the building block for conducting GPP logical operations on the Fibre Channel. The FC-PH interface to an Initiator or a Target is an architectural layer called the FC-4 level. This level is accomplished by GPP services.

A.4.1 Information unit

The manner in which Fibre Channel operations occur is for the logical element to request that a Task be performed with another Node. GPP services transforms TRB requests into Information Packets. GPP then transforms Information Packets into Information Units communicated to a destination Node using the services of the FC-3 and FC-2 levels of Fibre channel. The GPP FC-4 (GPP services) and the FC-2 level handle all details concerned with physically transmitting an Information Unit and notifying the receiving Node that an Information Unit has been received correctly. Figure 18 specifies the Information Unit used for GPP. See A.5 for a detailed specification of the Information Unit interface between the GPP services and FC-PH level FC-2.

ID	DATA BLOCK		F/M/L	SI	RO	M/O
	CAT	PAYLOAD				
I1 or T1	2	InfoPkt	Any Legal	H	N/A	M

LEGEND:

ID	- Short name for the information unit
CAT	- Data Category (unsolicited control only)
PAYLOAD	- Semantic meaning of the information unit
InfoPkt	- a GPP information packet
F/M/L	- First/Middle/Last Sequence of an Exchange
N/A	- Not applicable; any legal combination is acceptable
SI	- Sequence initiative (H - Hold; T - Transfer)
RO	- Relative Offset - not used in GPP
M/O	- Mandatory or optional information unit

Figure 18 - Information units

A.4.2 FC-PH Information Unit management

When an Information Packet is to be transferred, certain mandatory and optional information must be supplied along with the Information Packet contents. Annex S of FC-PH identifies the information which is supplied in addition to the Information Packet. In general, the information supplied is used to fill in FC-PH frame header fields.

The individual fields and their values specified for GPP are given below for Information Units carrying GPP Information Packets. Fibre Channel supports multiple protocols through the same service delivery interface. Therefore, GPP may share a single port with other protocols concurrently.

The Fibre Channel standard also specifies the content of Link_Control responses based on the content of received Device_Data frames. GPP does not specify any restriction on field use in Link_Control responses other than that required by the Fibre Channel standard.

The requirements and recommendations below provide high-level Exchange and Information Unit guidance for the GPP FC-4 level (GPP services) and the FC-2 level in the N_Port. To permit interoperability of GPP devices, this level of detail is required to properly construct the GPP services that operates between the GPP logical element and the FC-PH FC-2 functions in each N_Port. (See FC-PH, Annex S.)

A.4.3 GPP FC-PH responses and signals

FC-PH provides well-defined link level responses to Information Units received as part of an Exchange. Fibre Channel provides for FC-4-to-FC-4 and/or N_Port-to-N_Port notification of transmission status for Class 1 and Class 2 service.

Fibre Channel specifies the exact manner in which an Information Unit shall be transmitted, the normal response, and the abnormal responses. These specified responses shall be used for GPP. No additional response types, field values, or vendor unique values shall be specified. Logical responses to the content of an Information Packet are generated by the receiving logical element, where applicable, as new Information Packets sent in the opposite direction.

A.5 GPP services to FC-PH Information Unit services

Because this clause bridges between GPP Services and FC-PH function, the term FC-4, from Fibre Channel, applies. The term FC-4 will be used to retain continuity with other FC-PH standards. GPP FC-4 is equivalent to GPP Services in earlier clauses and earlier in this clause.

The term upper layer protocol (ULP) is another Fibre Channel specific term. In GPP the only ULP of interest is SCSI-3 GPP. The term ULP is used to retain continuity with other Fibre Channel standards. ULP in this clause is equivalent to a GPP logical element.

The services of FC-2 are used by a GPP FC-4 to cause an Information Unit transfer to another GPP FC-4. The service interface is patterned after the FC-PH service interface. The principal change is to remove multiple data categories per Information Unit which is not supported by GPP. The services used by a GPP device to indirectly invoke these functions are specified in clause 8.

The following service functions support GPP Information Unit transfers:

- FC_PH_SEQUENCE.request
- FC_PH_SEQUENCE_TAG.indication
- FC_PH_SEQUENCE.indication
- FC_PH_SEQUENCE.confirmation

The use of these services in a GPP Information Unit transfer are shown in figure 19.

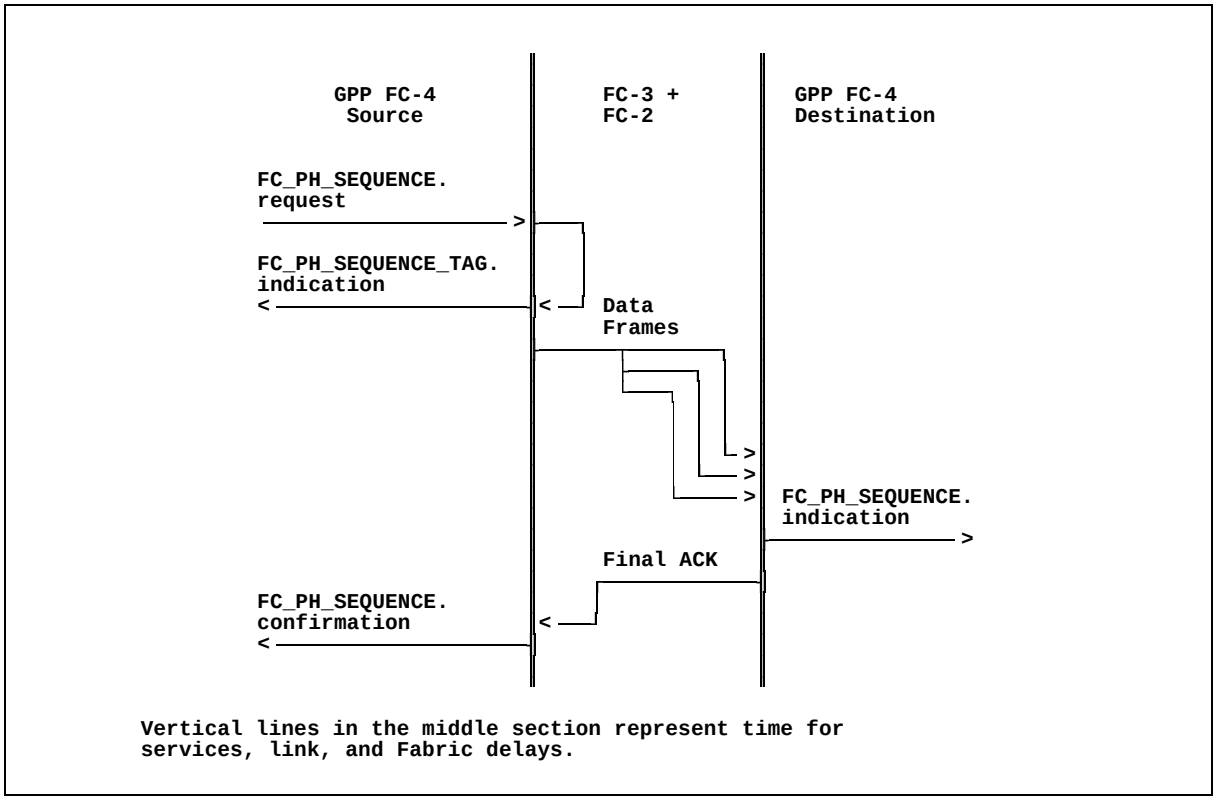


Figure 19 - FC-PH data transfer service primitives example

A.5.1 FC_PH_SEQUENCE.request

The FC_PH_SEQUENCE.request function specifies sufficient information for the transfer of one Information Packet from a local GPP ULP to a single peer GPP ULP (i.e., no broadcast or multicast function is provided).

All parameters in this function shall be provided. If a parameter is to be assigned by FC-PH, the invocation indicates that no value is supplied by the FC-4. The definition of a null value in the implementation is required.

A.5.1.1 Semantics

Figure 20 indicates the semantics of this request.

```
FC_PH_SEQUENCE.request (
                        Type,
                        Exchange_Tag,
                        Sequence_Tag,
                        Allowed_Class,
                        Routing_Bits,
                        D_ID,
                        S_ID,
                        First_Sequence,
                        Last_Sequence,
                        Sequence_Initiative,
                        Chained_Sequence,
                        Unidirectional Transmit,
                        Continue_Sequence_Condition,
                        Exchange_Error_Policy,
                        Relative_Offset_Present,
                        Exchange_Reassembly,
                        Data_Field_Control,
                        Expiration_Security_Header,
                        Network_Header,
                        Association_Header,
                        Device_Header,
                        Information_Category,
                        Block_Offset,
                        Data_Block )
```

Figure 20 - FC_PH_SEQUENCE.request semantics

A.5.1.2 When generated

This function is invoked by an FC-4 to request a GPP Information Packet transfer.

A.5.1.3 Effect of receipt

Upon receipt of a FC_PH_SEQUENCE.request function, the source FC-PH performs the following actions:

1 Indicates to the GPP FC-4 a unique Sequence_Tag using the FC_PH_SEQUENCE_TAG.indication function.

2 Validates the parameters and verifies that the requested operation is possible (e.g., checks against Login parameters, exchange initiative, etc).

3 If this is the first request for an Exchange, then FC-2 starts an Exchange with the first frame transmitted.

4 If this is a request that requires management of Exchange resources, then FC-2 provides an X_ID on the first frame of the first Sequence of the Exchange.

5 If this is a request to release Exchange resources, then FC-2 enables X_ID invalidated at the end of the Sequence.

6 If this is the last request for an Exchange, then FC-2 ends the Exchange with the last frame transmitted of the last Sequence.

7 Assigns a Sequence_ID to this transfer request.

8 Segments the Information Unit into frames for transmission. FC-PH determines frame boundaries.

9 Appends an appropriate frame header and trailer to each frame.

10 Transfers the frames to the destination.

11 Maintains the Exchange Status Block and Sequence Status Blocks.

12 Uses FC_PH_SEQUENCE.confirmation to notify GPP services that the Information Unit transfer has been successfully completed or abnormally terminated.

A.5.1.4 Function parameter usage

The Type parameter identifies the SCSI-3 GPP Upper Level Protocol (i.e., hex'09').

The Exchange_Tag provides a local identifier of the Exchange that includes this Information Unit. If the Exchange_Tag parameter is not supplied on the first Information Unit, the Exchange_Tag is assigned by FC-PH.

NOTE — The Exchange_Tag parameter is a unique identifier used to transfer an Information Unit. Since exchange resources are not shared between FC-4s, one or more exchanges are required to communicate with each other GPP device. These Exchanges are not required to be stable (i.e., when not in use they can be terminated).

Each GPP Exchange is required to operate in a unidirectional manner since sequence initiative is never transferred when processing a GPP Information Unit. Exchange Context and Sequence Context fields are managed by FC-PH as appropriate for the Exchange.

The Sequence_Tag parameter is not supplied for GPP invocations of this function. The Sequence_Tag is assigned by FC-PH and is indicated in the FC_PH_SEQUENCE_TAG.indication response.

The Allowed_Class parameter indicates the set of classes of service allowed for the Information Unit. An entire Information Unit is transmitted using one class from the allowed set. Only classes 1 and 2 are supported by GPP. Use of Class 3 service is outside the scope of GPP.

The Routing_Bits parameter indicates the routing for the Information Unit as FC-4 Device_Data (i.e., hex'0').

The D_ID parameter indicates the identifier of the desired destination of this Information Unit. The D_ID is one of or the only native N_Port identifier of the destination GPP device. The value chosen must be consistent with the level of multiple multipathing permitted between the two GPP devices.

The S_ID parameter indicates the native N_Port identifier of the source of the Information Unit. The S_ID is one of or the only native N_Port identifier of the source GPP device. The value chosen must be consistent with the level of multiple multipathing permitted between the two GPP devices.

NOTE — Since there is an FC-2 level associated with each N_Port, GPP services must pick the correct N_Port to request the Information Unit transfer based on the status of multipathing with the destination GPP device.

The First_Sequence parameter indicates when the Sequence is the first in a new Exchange (see 18.5). If the Exchange_Tag is null, this field is ignored (i.e., FC-PH establishes a new exchange and must set this field appropriately.

The Last_Sequence parameter indicates that the FC-4 requires that the Exchange be terminated.

The Sequence_Initiative parameter indicates when the Sequence Initiative is to be transferred to the Sequence Recipient of the Exchange. For GPP, this field always indicates that sequence initiative is HELD.

The Chained_Sequence parameter indicates that a Reply Sequence is expected and will be included within this Dedicated Connection. This function is not used by GPP since all Information Units in the reverse direction use an independent Exchange.

The Unidirectional Transmit parameter is not used by GPP since all Exchanges are unidirectional by definition without special port level controls.

The Continue_Sequence_Condition parameter indicates the expected time before the next Information unit is to be transferred. This field is not used by GPP.

The Exchange_Error_Policy parameter specifies the error recovery policy to be used in the Exchange. Its use is managed by FC-PH within the exchange. For GPP, the value shall be set to Abort, Discard Single Sequence (hex'1').

The Relative_Offset_Present parameter specifies whether relative offset processing is required for an Information Unit. For GPP, relative offset processing is not supported and this parameter shall be set to reflect that relative offsets are not used.

The Exchange_Reassembly parameter is reserved.

The Data_Field_Control parameter specifies the complete set of optional Headers that are be included with an Information Unit. For GPP, no optional Headers are specified. FC-PH may add optional headers as needed. If optional headers are added to an Information Unit, FC-2 shall set the DF_CTL field in the frame header to the appropriate value.

The Expiration_Security_Header parameter is not specified.

The Network_Header parameter is not specified.

The Association_Header parameter is not specified.

The Device_Header parameter is not specified.

The Information_Category parameter specifies the type of data contained in the Information Unit. For GPP, this value is always set to Unsolicited Control (i.e., hex'02').

The Block_Offset parameter is not used by GPP.

The Data_Block parameter specifies the Information Unit to be transferred. For GPP this may be a list of separate segments of bytes produced from the SG_List entries, ILE headers, Information Packet control prefix, etc (i.e., a gather list may be specified or a pointer to a contiguous block of bytes).

A.5.1.5 FC-PH managed parameters

The following fields in the FC-PH frame header are managed by FC-PH.

- Exchange Context
- Sequence Context
- End_Sequence
- E_C
- X-ID reassigned
- Invalidate X_ID
- Retransmitted Sequence
- Unidirectional Transmit
- Continue Sequence Condition
- Fill Data Bytes
- SEQ_ID
- DF_CTL

  If DF_CTL specifies an optional header, FC-PH supplies the appropriate optional header.
- Sequence count
- Originator Exchange ID
- Responder Exchange ID

A.5.2 FC_PH_SEQUENCE_TAG.indication

This function provides an indication of the Exchange_Tag and Sequence_Tag values used for identifying an Information Unit transfer. Both values are valid and unique until the confirmation is received by GPP services.

NOTE — This indication may not be unique, but only indicative of which Exchange was used to transmit the Information Unit. The Sequence_Tag value is unique within an Exchange at the time the Information unit is sent, but, over time, that value may be reused as well

A.5.2.1 Semantics

Figure 21 shows the semantics of this indication.

```
FC_PH_SEQUENCE_TAG.indication (
                                        Type,
                                        Exchange_Tag,
                                        Sequence_Tag )
```

Figure 21 - FC_PH_SEQUENCE_TAG.indication semantics

A.5.2.2 When generated

This function is generated in response to the FC_PH_SEQUENCE.request function.

A.5.2.3 Effect of receipt

The ULP may save the indication with the request and wait for the confirmation response.

A.5.2.4 FC-PH managed parameters

The Type parameter identifies the GPP Upper Level Protocol (i.e., hex'09').

The Exchange_Tag provides the local identifier of the Exchange that includes this Information Unit.

The Sequence_Tag provides a unique identifier of the Information Unit within the Exchange. The value may be reused after confirmation is sent to GPP services.

A.5.3 FC_PH_SEQUENCE.indication

This function indicates the receipt of a single Information Unit at the destination FC-PH for the GPP FC-4.

A.5.3.1 Semantics

Figure 22 shows the semantics of this indication.

```
FC_PH_SEQUENCE.indication (
                                    Type,
                                    Exchange_Tag,
                                    Class,
                                    Routing_Bits,
                                    D_ID,
                                    S_ID,
                                    First_Sequence,
                                    Last_Sequence,
                                    Chained_Sequence,
                                    Sequence_Initiative,
                                    Continue_Sequence_Condition,
                                    Exchange_Error_Policy,
                                    Relative_Offset_Present,
                                    Exchange_Reassembly,
                                    Data_Field_Control,
                                    Expiration_Security_Header,
                                    Network_Header,
                                    Association_Header,
                                    Device_Header,
                                    Information_Category,
                                    Block_Offset,
                                    Data_Block,
                                    Sequence_Valid )
```

Figure 22 - FC_PH_SEQUENCE.indication semantics

A.5.3.2 When generated

This function is generated after an Information Unit is successfully received. Since the Exchange_Error_Policy is Abort_Discard_Single_Sequence, the ordering of FC_PH_SEQUENCE.indications will occur in the same order as the Information Units arrive successfully, which is not necessarily the same as the order of transmission. GPP provides internal reordering Information Packets within a Task.

Upon receipt of portions of an Information Unit the destination FC-PH performs the following actions:

1 Maintains the state of the Class 1 Dedicated Connection, if applicable.

2 Assembles the received frames, recording any errors.

3 Maintains the Exchange Status Block and Sequence Status Blocks.

4 Issues ACK responses as required for the class of service used for the Exchange.

5 Uses the FC_PH_SEQUENCE.indication function to pass a completed Information Unit to the GPP FC-4. A partial Information Unit is not indicated to the destination FC-4. It is discarded based on the error policy selected above.

A.5.3.3 Effect of receipt

The receiving logical element may process a request to retrieve the payload of the Information Unit.

A.5.3.4 FC-PH managed parameters

The Type parameter identifies the GPP Upper Level Protocol (i.e., hex'09').

The Exchange_Tag provides a local (destination) identifier of the Exchange that received this Information Unit. This value can be the local RX_ID field value or the originators OX_ID field value.

The Class parameter indicates the class of service used for the Information Unit.

NOTE — This reflects the class of service for the Exchange and is likely to be a stable value, but that is not required since different classes of service may be used at different times.

The Routing_Bits parameter indicates the routing for the Information Unit as FC-4 Device_Data (i.e., hex'0').

The D_ID parameter indicates the identifier of the desired destination of this Information Unit.

The S_ID parameter indicates the native N_Port identifier of the source of the Information Unit.

The First_Sequence parameter indicates when the Sequence is the first in a new Exchange (see 18.5).

The Last_Sequence parameter indicates that the Exchange has been terminated.

The Chained_Sequence parameter indicates that a Reply Sequence is expected and will be included within this Dedicated Connection. This function is not used by GPP since all Information Units in the reverse direction use an independent Exchange.

The Sequence_Initiative parameter indicates when the Sequence Initiative is to be transferred to the Sequence Recipient of the Exchange. For GPP, this field always indicates that sequence initiative has been HELD by the Exchange originator.

The Continue_Sequence_Condition parameter indicates the expected time before the next Information Unit is to be transferred. This field is not used by GPP.

The Exchange_Error_Policy parameter specifies the error recovery policy to be used in the Exchange. For GPP, the value shall be set to Abort, Discard Single Sequence (hex'1').

The Relative_Offset_Present parameter specifies whether relative offset processing is required for an Information Unit. For GPP, relative offset processing is not supported and this parameter reflects that requirement.

The Exchange_Reassembly parameter is reserved.

The Data_Field_Control parameter specifies the complete set of optional Headers that accompany the Information Unit. For GPP, no optional Headers are specified. FC-PH may add optional parameters as needed. Use of optional headers by the GPP FC-4 is outside the scope of GPP.

The Expiration_Security_Header parameter, if indicated by the Data_Field_Control parameter value.

The Network_Header parameter, if indicated by the Data_Field_Control parameter value.

The Association_Header parameter, if indicated by the Data_Field_Control parameter value.

The Device_Header parameter, if indicated by the Data_Field_Control parameter value.

The Information_Category parameter specifies the type of data contained in the Information Unit. For GPP, this value is always set to Unsolicited Control (i.e., hex'02').

The Block_Offset parameter is not used by GPP.

The Data_Block parameter specifies the Information Unit received.

NOTE — The means for GPP services to preallocate a storage area for an Information Unit is an implementation decision.

A.5.4 FC_PH_SEQUENCE.confirmation

This function provides an appropriate response to a FC_PH_SEQUENCE.request indicating the success or failure of the request.

A.5.4.1 Semantics

Figure 23 shows the semantics of this confirmation.

```
FC_PH_SEQUENCE.confirmation (
                                Type,
                                Exchange_Tag,
                                Sequence_Tag,
                                Transmission_Status,
                                Reject_Reason )
```

Figure 23 - FC_PH_SEQUENCE.confirmation semantics

A.5.4.2 When generated

This function is generated upon the completion of an attempt to transmit the Sequence.

A.5.4.3 Effect of receipt

The FC-4 may prepare another request for FC-PH for the same port.

NOTE — The number of concurrent requests to FC-2 is an implementation decision.

A.5.4.4 FC-PH managed parameters

The Type parameter identifies the GPP Upper Level Protocol (i.e., hex'09').

The Exchange_Tag parameter provides the local identifier of the Exchange that includes this Information Unit.

The Sequence_Tag parameter provides a unique identifier of the Information Unit within the Exchange.

The Transmission_Status parameter will indicate status as one of the following:

- Successful - Sequence delivered completely to Recipient

- Unsuccessful - Sequence was not delivered completely due to abort or frame transfer error.

- Stopped_by_Recipient - Recipient stopped the Information Unit transfer as indicated in an ACK response during the Information Unit transfer.

- Rejected_Request - The Sequence was not sent by the Initiator due to the specified Reject_Reason.

- Rejected_by_Fabric - Reject frame received from Fabric

- Rejected_by_N_Port - Reject frame received from N_Port

When the Transmission_Status parameter value is Rejected_Request, Rejected_by_Fabric, or Rejected_by_N_Port, the Reject_Reason parameter indicates one of the Reject reason codes specified in FC-PH, 20.3.3.3.

A.6 FC-PH events

FC-PH has two asynchronous events:

- the unexpected disconnect event

- the unsuccessful Information Unit transfer event.

Each event establishes a condition which causes GPP services to perform certain recovery actions.

A.6.1 FC-PH unexpected disconnect event

GPP devices normally expect FC-PH defined primitives to begin on its receive fibre after one of the following occurs:

1 after the successful transfer of each frame for an Information Unit.

2 after one of the FC-PH link protocol responses.

All other occurrences of an unexpected state cause the GPP device to report an unexpected disconnect event to the sending GPP services. The receiving GPP services shall handle this as an unsuccessful Information Unit transfer condition.

A.6.2 FC-PH unsuccessful Information Unit transfer event

If a complete Information Unit is not received according to FC-PH requirements, the receiving GPP device shall discard any portion of the Information Unit received and indicate an unsuccessful Information Unit transfer event. The receiving GPP services shall handle this as an unsuccessful Information Unit transfer condition.

A.6.3 FC-PH unsuccessful Information Packet transfer condition

While processing an unsuccessful Information Unit transfer condition, the sending GPP device:

1 may attempt to retransmit the Information Unit in error up to a limit determined by the sending FC-4.

2 if operating as the Target for the Task, the sending GPP services shall notify its ULP and the Task shall be terminated after the specified number of retries by clearing all pending TRBs for the Task and preparing sense data.

3 if operating as the Initiator for a Task, the sending GPP device shall abort the Task using the ABORT TASK Task Management function. It is recommended that the Initiator then request sense data from the Target if an autosense ILE is not provided by the Target for the Task.

Because of the full-duplex nature of Fibre Channel, a Task completion Information Packet may arrive overlapped with transmission of the ABORT TASK Task Management function.

ANNEX B
(informative)

FC-PH information transfer for GPP

An active serial interface consists of a continuous stream of encoded words transmitted along one fibre between two ends of a link. This stream of transmission words is

- between two N_Ports in a point-to-point topology.
- between an N_Port and its associated F_Port in a switched point-to-point topology.
- between two E_Ports in a switched point-to-point topology between two Fabric elements.
- between two NL_Ports or between an NL_Port and an FL_Port in an arbitrated loop topology.

Fibre Channel is a full-duplex protocol and therefore, there are two such fibres: one into the port and one out of the port. In GPP terms the receiving port is not aware that it is about to receive an Information Packet. The sending GPP port may not be aware of the present state of the destination GPP port.

The GPP information transfer protocol assumes delivery of complete Information Packets and recovers on that boundary when that assumption proves incorrect. That is, error free Information Packet delivery is not guaranteed at the time of transmission. The Exchange protocol provides for positive responses from the opposite end of the link at some time after transmission (classes 1 and 2). The GPP Exchange protocol, as well as other FC-PH mappings, considers a Fabric a transparent element in the protocol.

For any point-to-point topology, the source has complete control of when an Information Packet is transmitted. The destination also has complete control of when an Information Packet is transmitted on the reverse fibre.

For each Information Packet which contains a logical error, but which is otherwise received error free by GPP services, the receiving FC-PH port sends appropriate acknowledgement response(s) for the Information Unit. The receiving GPP logical unit does not process this Information Packet beyond the point of the first detected logical error. Any GPP logical error detected by a logical element is reported using an Information Packet sent in the opposite direction.

For each Information Packet received with a service delivery subsystem error for the Information Unit, such as a CRC error, the receiving GPP FC-PH port and GPP services, in particular, does present any portion of the Information Packet to a GPP logical element. The N_Port does not process this Information Unit further and does not retain it. An appropriate FC-PH response is sent to the transmitter of the Information Unit (classes 1 and 2).

B.1 Introduction to the GPP FC-4 Exchange protocol

Two GPP devices that require interaction must have a means to communicate:

- The lowest level Fibre Channel protocol involves individual frame transmission and reporting of buffer-to-buffer and end-to-end flow control. This level of operation is transparent to GPP services.
- The next level of FC-PH protocol is Sequence construction, transmission, and reassembly. This level of operation is transparent to GPP services.
- The next level of FC-PH protocol is Information Unit management. In GPP, an Information Packet is equivalent to one Information Unit which is equivalent to one FC-PH Sequence. A GPP services provides the set of bytes in the form of an Information Packet to be transmitted by the FC-2 level, and the receiving FC-4 provides a

reassembled Information Packet to the GPP logical element. The content of the Information Unit payload is not known by, and therefore is not interpreted by, FC-PH or the GPP FC-4 level. The actual Information Unit transmission details are managed by FC-PH. The GPP FC-4 is the lowest level of interaction between a GPP logical element and FC-PH.

- The next level protocol in FC-PH is the Exchange which consists of the transfer of one or more Information Units and the corresponding FC-PH link level responses. An Exchange is an administrative construct that provides permission to send Information Units. The actual management details are performed by the FC-4 level and FC-PH. GPP services can start and stop Exchanges, but otherwise, it cannot manage them.

  An Exchange permits transfer of Information Units in only one direction at a time by the owner of Sequence Initiative. To be able to effectively use the entire bandwidth between two N_Ports, two or more Exchanges are required (assuming sufficient total workload). GPP specifies the use of two Exchanges per pair of N_Ports in a logical system, each with Sequence Initiative in the opposite direction. Additional Exchanges may be active if required. See figure 24.

  There is no FC-PH tie between these Exchanges, and there is no GPP tie between these Exchanges. The Exchanges are for the owners of Exchange Sequence Initiative to use as needed. Since only one Information Unit can be transmitted at a time per N_Port, use of multiple Exchanges per N_Port is not required.

  Information Units may be streamed up to the N_Port login limit for Open Sequences per Exchange. If Sequence streaming is not permitted in an implementation, additional Exchanges may be established to eliminate the turn-around delay between Sequences in the same Exchange.

  The SCSI concept of a Task for GPP is not related to an Exchange. The Exchange is the required FC-PH vehicle used to transmit Information Units between two N_Ports. Two consecutive Information Units in the same Exchange between two N_Ports may have no relation to each other, except that the logical unit(s) and Initiator functions reside inside the respective N_Ports. Each Target N_Port may have one or more open Exchanges per Initiator N_Port in its logical system.

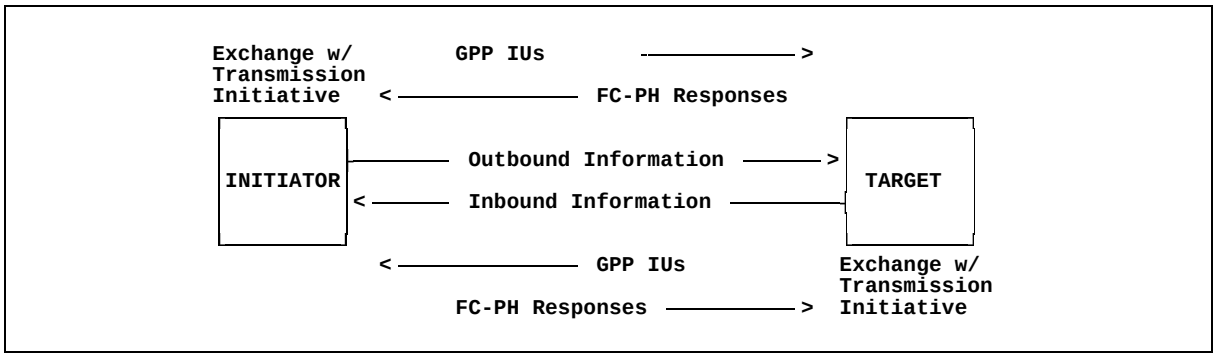


Figure 24 - GPP Exchange Protocol

If it is necessary for one of the N_Ports to terminate some or all of its Exchanges, it may do so. Since Sequence Initiative is not transferred within one Exchange, the N_Port must establish at least one Exchange to continue or start an Task with a destination N_Port.

Also, since there is no affinity between a GPP Task and any FC-PH Exchange, multiple N_Port pairs may be used to conduct a single Task when multipathing is supported. This permits the use of multiple ports and even multiple Fabrics for highly available systems.

- The highest level of Fibre Channel protocol which affects GPP is N_Port login. This protocol is specified in FC-PH and executed outside GPP, but it affects GPP. Without successful N_Port login, Exchanges for the purpose of transferring GPP Information Packets cannot be established with another N_Port. Other activities,

like Fabric login or arbitrated loop initialization, must occur before normal GPP Information Packet transfer can begin.

- GPP adds no FC-PH protocol over that minimally required by FC-PH to establish communication and maintain reasonable communication between N_Ports for any FC-PH supported topology. Therefore, GPP can be added to any extant FC-PH environment with no change in equipment or procedures.

B.2 GPP FC-PH Exchange protocol

This subclause specifies how GPP uses Fibre Channel and does not reflect the full capabilities of Fibre Channel. The Exchange protocol is composed of the following elements:

- Information Unit
- Exchanges

B.2.1 FC-PH Information Units

A GPP Information Packet maps one-to-one with an FC-PH Information Unit. Once Fabric and N_Port login are complete and an Exchange is established in each direction between two communicating N_Ports, Information Packets may be transferred between them using FC-PH Information Units. The first Information Unit transferred can be used to establish this Exchange.

Information Units may arrive at a destination in an order different than the order transmitted. This is a potential outcome of using Class 2 service with FC-PH. Additionally, if an Information Unit is received in error, it is discarded and the sender is notified to retransmit the Information Unit. This retransmission may also cause Information Units to be sent to a destination ULP out of order. GPP provides an internal sequencing mechanism to maintain the order of Information Packets within each Task. This Sequence control is not part of the FC-PH protocol since GPP is designed to use other interfaces which may have no mechanisms for managing or detecting order of transmission.

Also, GPP permits multiple paths to be used to transmit Information Packets for the same Task. There is no ordering information carried in FC-PH between communicating N_Port pairs for this case. Also, once multiple pathing is in effect, the path pairs may be in different Fabrics on independent arbitrated loops.

B.2.2 FC-PH Exchanges

An Exchange establishes a logical connection between two N_Ports for the purpose of transmitting Information Units. An Exchange is established by a sending N_Port after N_Port login. The Originator of an Exchange identifies its Exchange by the concatenation of S_ID || D_ID || OX_ID (selected by S_ID). The Responder assigns a Responder Exchange identifier (RX_ID) to the Exchange.

In the reverse direction, a second Exchange is established with the Exchange identification D_ID || S_ID || OX_ID (selected by D_ID). The Responder assigns a Responder Exchange identifier (RX_ID) to this second Exchange.

In normal practice, the Exchanges may be established while attempting to negotiate a packet transfer request agreement using Information Packets containing the PACKET TRANSFER REQUEST message (PTR). This negotiation is normally performed only once per N_Port login. The negotiation is terminated when either N_Port logs out (implicitly or explicitly) with the other or a new negotiation occurs.

For example, a GPP Initiator N_Port is assigned a Native Address Identifier of 777777h and a GPP Target N_Port is assigned a Native Address Identifier of 111111h during their respective Fabric login procedures. Figure 25 shows

a portion of a Task which establishes a pair of unidirectional Exchanges used with GPP. If the PTR negotiation is not used, the pair of Exchanges can be established as part of any Task which requires Information Packets in both directions (the first Information Unit in each direction can establish an Exchange with the proper Sequence Initiative). The Exchanges may be established by any other FC-PH defined means also. Refer also to figure 25.

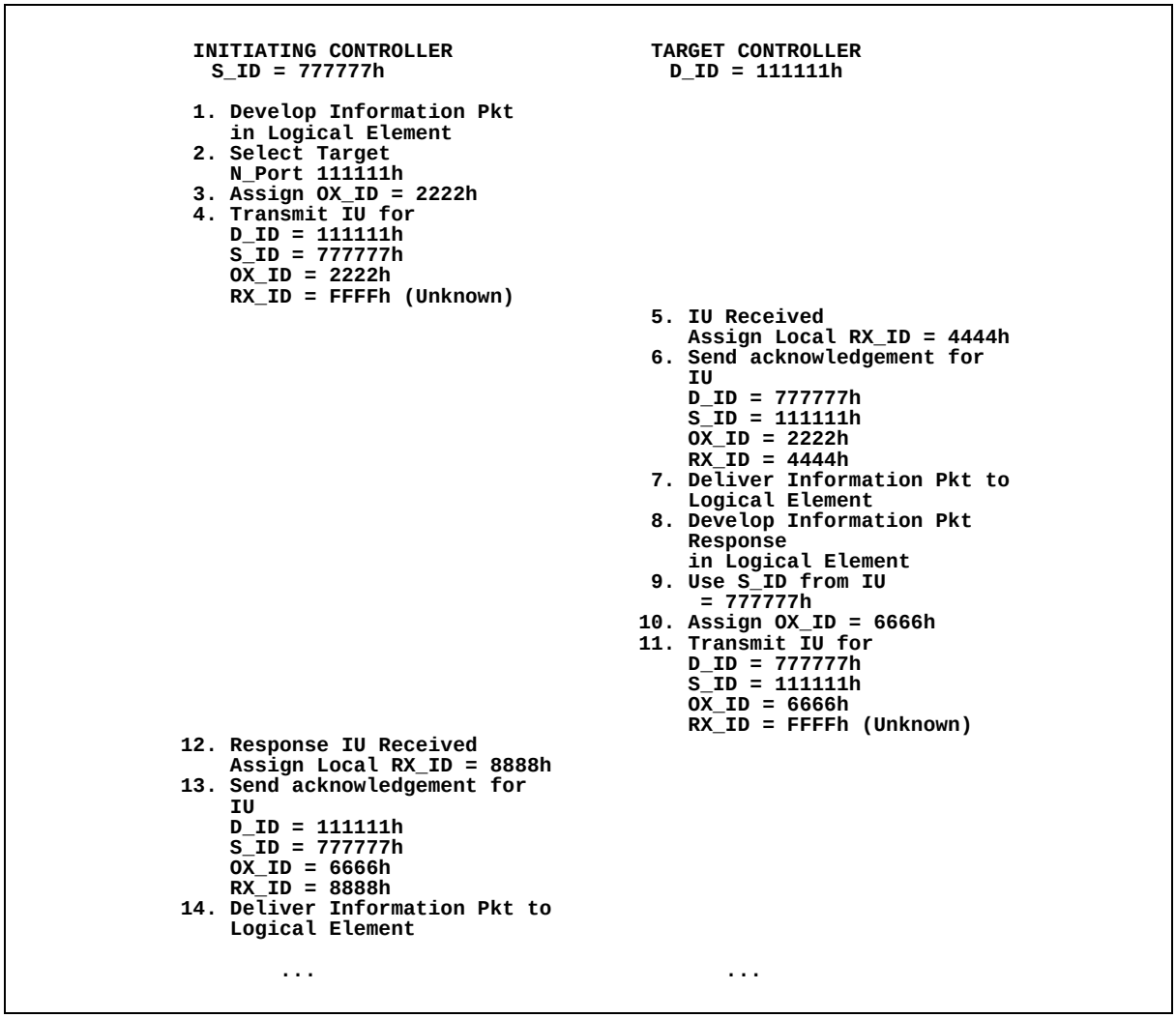


Figure 25 - Example of Establishing Bi-directional Exchanges

Since all GPP information is transferred in Information Packets. normal operation can now proceed. The Initiator develops Information Packets which start Tasks and the Target and the associated logical unit acts on the Task and prepares the appropriate responses as required by the SCSI-3 Architecture Model and the SCSI-3 command sets.

B.2.3 Transport layer independence

Since all Information Packets are fully self-identifying, relative to the Task with which they are associated, there is no connection between these two pairs of Exchanges or to the Tasks. Additional Exchanges may be set up with the same or a different set of N_Ports between the same Initiator and Target. Each Node may use any available GPP Exchange between them to transfer an Information Unit to the other.

This lack of dependence on the physical transport system permits the multiple port option of GPP to be used with any type of Task spanning multiple N_Ports per logical element within a single Fabric or spanning two or more Fabrics or with other service delivery subsystems. See figure 26.

NOTE — An alternate set of ports could use the SCSI-3 Parallel Interface or any of the other interfaces in these appendices.

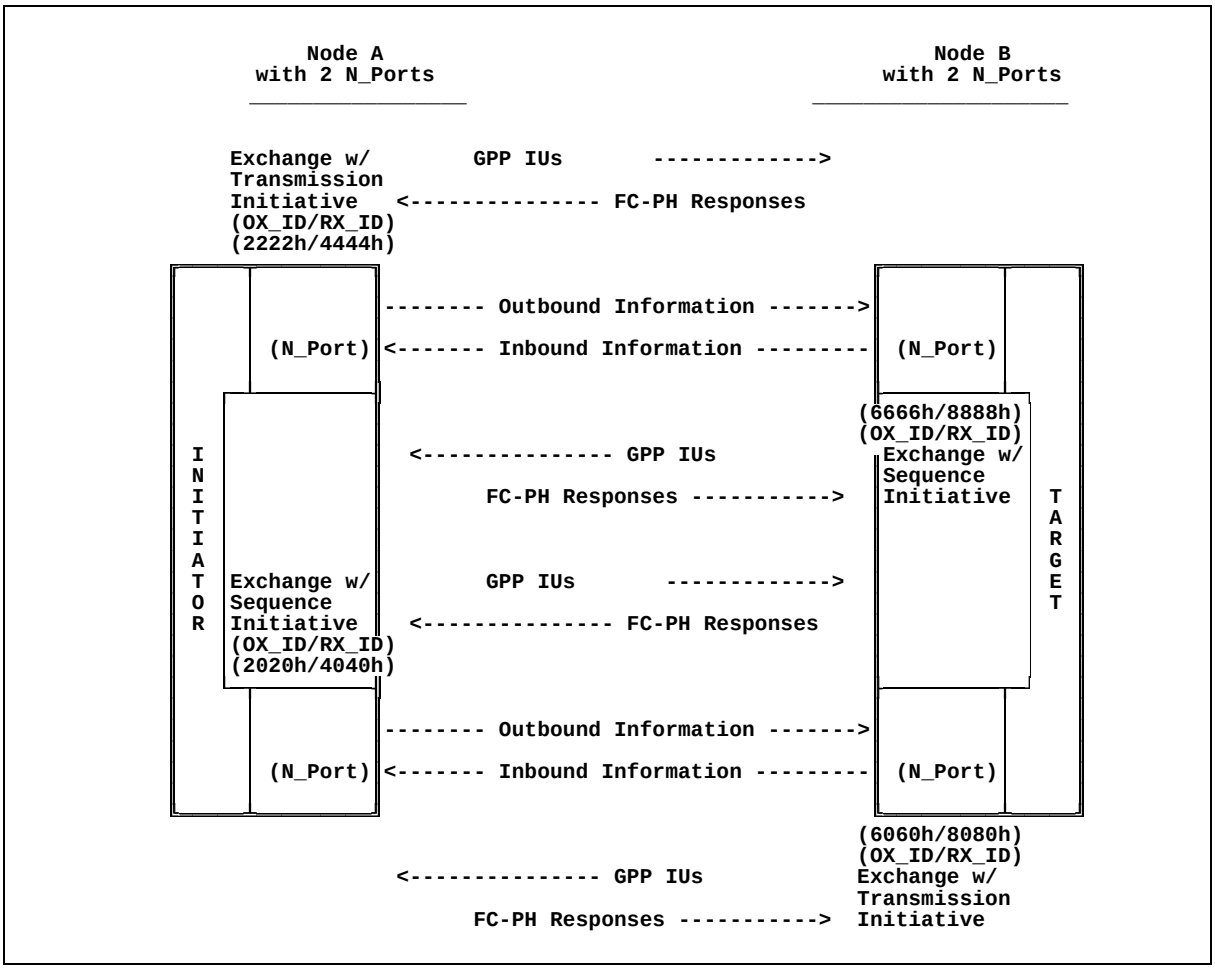


Figure 26 - GPP Logical System with Multiple Ports

ANNEX C
(normative)

SPI usage for GPP

This annex specifies the parallel service delivery subsystem available to GPP. Each service delivery subsystem standard defines the physical requirements for interconnection. For specific details on the SCSI-3 parallel bus, see the SCSI-3 Parallel Interface (SPI) standard. See Clause 2 for the complete titles and numbers for these standards.

GPP requires a homogeneous bus width for all GPP devices attached to a single SPI bus. The assumption is that all GPP devices will implement the 2-byte wide version of SPI for performance reasons. In addition, they probably will not have backward compatibility with the SCSI-3 interlocked protocol (SIP). This requirement eliminates the wide data transfer negotiation and, it simplifies and speeds up connections since the width becomes a constant for GPP. No intermix of bus widths is permitted since no wide data transfer negotiation is defined. If a GPP device implements the interlocked protocol, the device may switch to that mode of operation and use the SIP message system to negotiate bus widths according to that protocol. However, they have no effect on GPP operations.

This annex and annex D specify the use of SPI as a transport layer for GPP. A logical system may be constructed using any of the service delivery subsystems in GPP or a combination of two or more.

C.1 SPI physical description

The implementer shall select from the busses, shown in figure 27, from the SPI standard. The "X" identifies valid GPP options. Single ended and differential implementations are mutually exclusive. GPP does not support intermixing 8-bit, 16-bit, and 32-bit interfaces. All GPP devices that are principally initiating controllers shall supply terminator power as required for termination option A2.

Electrical Type	Connector Type			
	Shielded		Non-Shielded	
Single Ended (6 M)	A1	A2	A1	A2
8-Bit (A Cable)				
Termination A1	-	-	-	-
Termination A2	X	-	X	-
16-Bit (P Cable)				
Termination A1	-	-	-	-
Termination A2	X	-	X	-
32-Bit (P+Q Cable)				
Termination A1	-	-	-	-
Termination A2	X	-	X	-
Bus Width Intermix				
Termination A1	-	-	-	-
Termination A2	-	-	-	-
Differential (25 M)				
8-Bit (A Cable)				
5 MHZ	X	-	X	-
Fast Synchronous	X	-	X	-
16-Bit (P Cable)				
5 MHZ	X	-	X	-
Fast Synchronous	X	-	X	-
32-Bit (P+Q Cable)				
5 MHZ	X	-	X	-
Fast Synchronous	X	-	X	-
Bus Width Intermix				
5 MHZ	-	-	-	-
Fast Synchronous	-	-	-	-

Figure 27 - SPI transport layer options

C.2 SPI specific messages

This subclause specifies service delivery subsystem dependent messages that are required when GPP operates on the SCSI-3 Parallel Interface. All of the messages are extended messages.

The extended message codes are listed in table 23. This table is an extension of table 5. The extended message arguments are specified with each extended message description. For extended messages, the message code is followed by a variable length set of zero or more parameter bytes. The function to be performed is determined by the value in the Extended Message Code field.

Table 23 - SPI Extended Message Codes

Code Value	**Name (Ordered by Extended Message Value)**
02h	**INVALID BUS PHASE**
03h	**PARITY ERROR**
01h	**SYNCHRONOUS DATA TRANSFER REQUEST**
	Name (Ordered by Extended Message Code)
01h	**SYNCHRONOUS DATA TRANSFER REQUEST**
02h	**INVALID BUS PHASE**
03h	**PARITY ERROR**

SPI specific messages are defined below. The requirements for message implementation are given in table 24.

Table 24 - SPI specific message codes

Code	Initiator S/R	Target S/R	Message Name	Direction	Singular
EXTD	**M/M**	**M/M**	**INVALID BUS PHASE**	**In/Out**	**Yes**
EXTD	**M/M**	**M/M**	**PARITY ERROR**	**In/Out**	**Yes**
EXTD	**M/M**	**M/M**	**SYNCHRONOUS DATA TRANSFER REQUEST**	**In/Out**	**Yes**

KEY:

EXTD = Extended Message Code 01h
In = Target to Initiator
Out = Initiator to Target
M = Mandatory support
No = Ma be other ILEs in Packet
N = Not Sent or Not Received as appropriate by column
O = Optional support
S/R = Send / Receive
Yes = Only ILE in a Packet

C.2.1 INVALID BUS PHASE DETECTED

The INVALID BUS PHASE DETECTED message (table 25) is sent from either logical element to the other to inform it that an illegal or reserved bus phase was detected on the SPI.

The Information Packet containing an INVALID BUS PHASE DETECTED message shall contain only one interface logical element, a message interface logical element.

Table 25 - INVALID BUS PHASE DETECTED message format (SPI)

Byte	Value	Description			
0	01h	Extended Message Format Code			
1	02h	Extended Message Length			
2	02h	Extended Message Code			
3	xxh	Bits 7 - 3 Reserved	Bit 2 C/D	Bit 1 I/O	Bit 0 Msg

The C/D bit shall contain 1 if the phase where the phase error was detected has the C/D signal set to true. Otherwise, the C/D bit shall be set to zero.

The I/O bit shall contain 1 if the phase where the phase error was detected has the I/O signal set to true. Otherwise, the I/O bit shall be set to zero.

The Msg bit shall contain 1 if the phase where the phase error was detected has the MSG signal set to true. Otherwise, the Msg bit shall be set to zero.

These three fields identify the information transfer phase which was detected in error.

C.2.2 PARITY ERROR

The PARITY ERROR message (table 26) is sent from either logical element to the other to inform it that a parity error was detected during an Information Packet transfer on a SPI interface.

The Information Packet containing a PARITY ERROR message shall contain only one interface logical element, a message interface logical element.

Table 26 - PARITY ERROR message format (SPI)

Byte	Value	Description			
0	01h	Extended Message Format Code			
1	02h	Extended Message Length			
2	03h	Extended Message Code			
3	xxh	Bits 7 - 3 Reserved	Bit 2 C/D	Bit 1 I/O	Bit 0 Msg

The C/D bit shall contain 1 if the invalid phase detected has the C/D signal set to true. Otherwise, the C/D bit shall be set to zero.

The I/O bit shall contain 1 if the invalid phase detected has the I/O signal set to true. Otherwise, the I/O bit shall be set to zero.

The Msg bit shall contain 1 if the invalid phase detected has the MSG signal set to true. Otherwise, the Msg bit shall be set to zero.

These three fields identify the information transfer phase where the parity error was detected.

If the error is detected during transfer of the Information Packet prefix fields, the receiving logical element shall assert the ATN signal during the same phase to interlock the error with the current I/P process. The receiving logical element shall form an Information Packet using the information from the Information Packet prefix control fields, as received and include a PARITY ERROR message as the only interface logical element in the Information Packet.

C.2.3 SYNCHRONOUS DATA TRANSFER REQUEST

A SYNCHRONOUS DATA TRANSFER REQUEST (SDTR) message (table 27) exchange shall be initiated by any GPP device whenever a previously-arranged data transfer agreement may have become invalid. The default agreement shall be asynchronous information transfer mode.

The Information Packet containing an SYNCHRONOUS DATA TRANSFER REQUEST message shall contain only one interface logical element, a message interface logical element.

The SDTR message exchange establishes the permissible transfer periods and the REQ/ACK offsets for all logical units on the two devices. This agreement only applies to the DATA OUT information transfer phase used by GPP.

Table 27 - SYNCHRONOUS DATA TRANSFER REQUEST message format (SPI)

Byte	Value	Description
0	01h	Extended Message Format Code
1	03h	Extended Message Length
2	01h	Extended Message Code
3	xxh	Transfer Period (m times 4 nanoseconds)
4	xxh	REQ/ACK Offset

The transfer period field is the minimum time allowed between leading edges of successive REQ pulses and of successive ACK pulses to meet the device requirements for successful reception of data.

The REQ/ACK offset field is the maximum number of REQ pulses allowed to be outstanding before the leading edge of its corresponding ACK pulse is received at the receiving GPP device. This value is chosen to prevent overflow conditions in the receiving buffer and offset counter. A REQ/ACK offset value of zero shall indicate asynchronous data transfer mode; a value of FFh shall indicate unlimited REQ/ACK offset.

The synchronous data transfer agreement becomes invalid, and reverts to asynchronous data transfer mode, after any condition which may leave the data transfer agreement in an indeterminate state such as:

- after a reset event
- after a DEVICE RESET message and
- after a power cycle.

In addition, an GPP device may initiate an SDTR message exchange whenever it is appropriate to negotiate a new synchronous data transfer agreement.

The originator (the GPP device that sends the first SDTR message) sets its values according to the rules above to permit it to receive data successfully. If the responding GPP device can also receive data successfully with these values (or smaller transfer periods or larger REQ/ACK offsets or both), it returns the same values in its SDTR message. If the responding GPP device requires a larger transfer period, a smaller REQ/ACK offset, or both to receive data successfully, it substitutes values in its SDTR message as required, returning unchanged any value not required to be changed. Each device when transmitting data shall respect the limits set by each successful negotiation, but it is permitted to transfer data with larger transfer periods, smaller REQ/ACK offsets, or both than agreed to in the most recent successful negotiation.

Successful negotiation occurs upon completion of an exchange of two identical SDTR messages, including an agreement where the REQ/ACK offset is zero. Successful negotiation also implies that the agreement is retained by both GPP devices until reset or renegotiated. The synchronous negotiation takes effect on the next Information Packet transfer following the exchange of one SDTR message in each direction with identical values in all fields.

NOTE — SDTR messages are singular, so a disconnect must occur before the next information packet transfer.

A minimum of one message in each direction is required to confirm a negotiation. If an Initiator starts with one set of values, the Target changes the values in its response and the Initiator agrees with the new values it must respond by sending the identical message back to the Target. A total of three messages shall be transferred if the Target changes the values in the first message:

- two by the initiating GPP device

- one by the responding GPP device.

C.3 GPP services to SPI information transfer services

The GPP protocol must be distinguished from the SIP protocol and the address assignment protocol of SPI. To do so, GPP uses a modified sequence of phases to accomplish this.

GPP services forms Information Packets from information supplied by a logical element as specified in clause 8. The services of the service delivery interface are a separate set of services. The SPI services are similar to the FC-PH services in that they are aware of the low-level physical protocol. GPP services assumes a similar interface to SPI. The function is not nearly as rich as with FC-PH, but identifying a distinct service set continues the parallelism with the FC-PH annexes. Any reference to ULP refers to a GPP logical element.

The SPI services are used by GPP services to cause an Information Packet transfer to another GPP device using the SCSI-3 Parallel Interface (SPI). The service interface is patterned after the FC-PH service interface. The services used by a GPP device to indirectly invoke these functions are specified in clause 8.

The following service functions support GPP Information Packet transfers for SPI:

- SPI_SEQUENCE.request

- SPI_SEQUENCE_TAG.indication
- SPI_SEQUENCE.indication
- SPI_SEQUENCE.confirmation

The use of these services in a GPP Information Packet transfer are shown in figure 28.

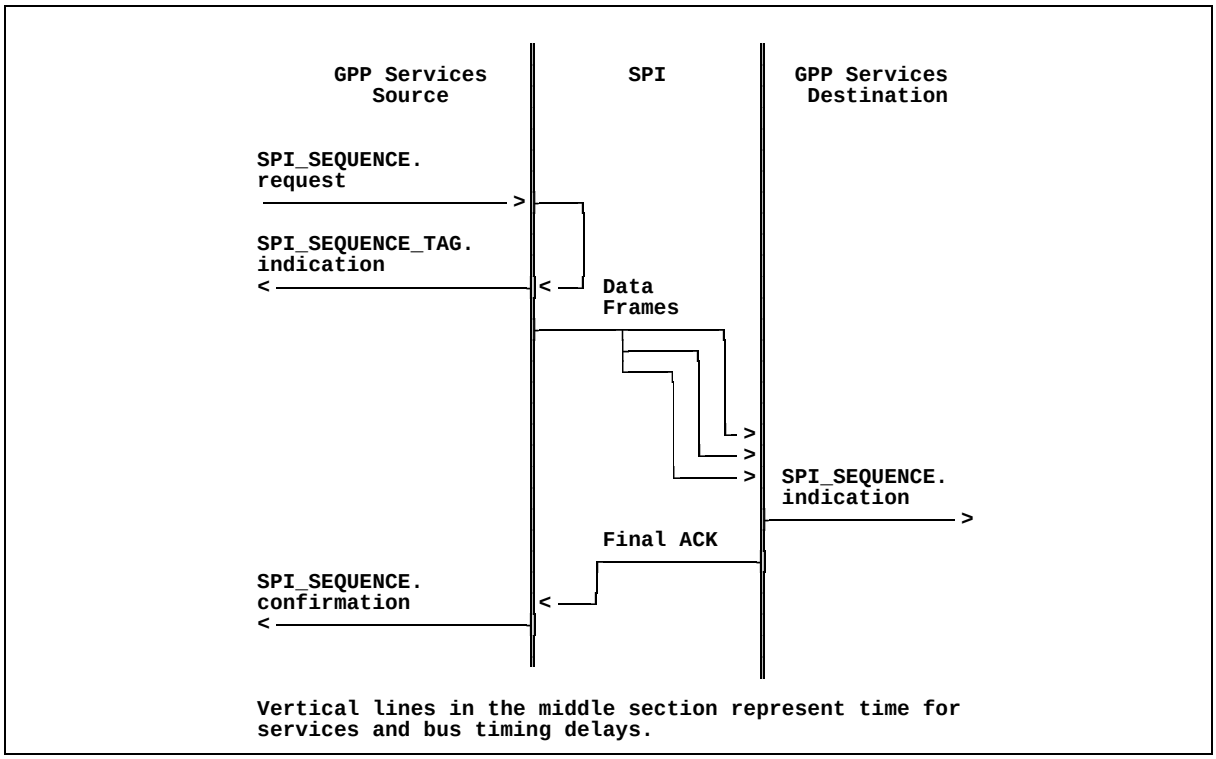


Figure 28 - FC-PH data transfer service primitives example

C.3.1 SPI_SEQUENCE.request

The SPI_SEQUENCE.request function specifies sufficient information for the transfer of one Information Packet from the local GPP services to a single peer GPP services (i.e., no broadcast or multicast functions are available).

All parameters in this function shall be provided. If a parameter is to be assigned to SPI, the invocation indicates that no value is supplied by the GPP services.

NOTE — Each implementation must select the not-supplied value. It is not specified by GPP services.

C.3.1.1 Semantics

Figure 29 shows the semantics of this request.

```
SPI_SEQUENCE.request (
                        Sequence_Tag,
                        D_ID,
                        S_ID,
                        Data_Block )
```

Figure 29 - SPI_SEQUENCE.request semantics

C.3.1.2 When generated

This function is invoked by GPP services to request an Information Packet transfer.

C.3.1.3 Effect of receipt

Upon receipt of a SPI_SEQUENCE.request function, the source SPI service performs the following actions:

1 Indicates to GPP services a unique Sequence_Tag using the SPI_SEQUENCE_TAG.indication function.

2 Validates the parameters and verifies that the requested operation is possible (e.g., checks against negotiated parameters, etc).

3 Transfers the Information Packet to the destination.

4 Uses SPI_SEQUENCE.confirmation to notify GPP services that the Information Packet transfer has been successfully completed or abnormally terminated.

C.3.1.4 Function parameter usage

The Sequence_Tag parameter is not supplied by GPP services when invoking this function. The Sequence_Tag is assigned by SPI services and is indicated in the SPI_SEQUENCE_TAG.indication.

The D_ID parameter indicates the identifier of the desired destination of this Information Packet. The D_ID is one of or the only SCSI address of the destination GPP device. The value chosen must be consistent with the level of multipathing permitted between the two GPP devices.

The S_ID parameter indicates the address identifier of the source of the Information Packet. The S_ID is one of or the only SCSI address of the source GPP device. The value chosen must be consistent with the service delivery interface state relative to SPI and the level of multipathing permitted between the two GPP devices. Selecting the service delivery interface shall be the responsibility of GPP services. SPI services shall supply only single port management services.

The Data_Block parameter specifies the Information Packet to be transferred. The Data block may be a single entry with a pointer and length or a gather list in the correct order to form an Information Packet. Each implementation must select whether a single entry or a list is permissible.

NOTE — The gather list option should provide faster performance since data does not need to be moved an extra time to form a contiguous Information Packet.

C.3.1.5 SPI managed parameters

The following fields in the Information Packet header are managed by SPI services.

- Initiator Port Number
- Original Initiator GPP Address
- Original Target GPP Address
- Target Port Number

C.3.2 SPI_SEQUENCE_TAG.indication

This function provides an indication of the Sequence_Tag value used for identifying an information unit transfer.

C.3.2.1 Semantics

Figure 30 shows the semantics of this indication.

```
SPI_SEQUENCE_TAG.indication (
                                    Sequence_Tag )
```

Figure 30 - SPI_SEQUENCE_TAG.indication semantics

C.3.2.2 When generated

This function is generated in response to the SPI_SEQUENCE.request function.

C.3.2.3 Effect of receipt

The Sequence_Tag value provides a mechanism for GPP services to track a request and match it with the final confirmation.

C.3.2.4 SPI managed parameters

The SPI_SEQUENCE_TAG.indication provides a unique identifier of the information unit within the Exchange.

C.3.3 SPI_SEQUENCE.indication

This function indicates the receipt of a single information unit at the destination SPI for the GPP FC-4.

C.3.3.1 Semantics

Figure 31 shows the semantics of this indication.

```
SPI_SEQUENCE.indication (
                                D_ID,
                                S_ID,
                                Data_Block,
                                Sequence_Valid )
```

Figure 31 - SPI_SEQUENCE.indication semantics

C.3.3.2 When generated

This function is generated after an information unit is successfully received. Since the general policy for GPP is to discard any single Information Packet received in error, the ordering of SPI_SEQUENCE.indications will occur in the same order as the information units arrive successfully, which is not necessarily the order of transmission. GPP provides internal reordering of Information Packets within a task.

Upon receipt of an Information Packet the destination SPI performs the following actions:

1 Uses the SPI_SEQUENCE.indication function to pass a completed Information Unit to GPP services.

C.3.3.3 Effect of receipt

Receipt of an Information Packet is signalled to the destination GPP services for further processing.

C.3.3.4 SPI managed parameters

The D_ID parameter indicates the identifier associated with the SPI service delivery interface where this Information Packet was received.

NOTE — A SPI service delivery interface may respond to multiple SCSI addresses. This value must be valid for the receiving port to respond to SELECTION phase.

The S_ID parameter indicates the SPI address of the source of the Information Packet.

NOTE — This value is determined during the SELECTION phase preceding the Information Packet transfer(s).

The Data_Block parameter specifies the Information Unit received. This is either a single entry containing a pointer and length or it is a gather list with a set of such entries.

NOTE — GPP does not specify the means of receiving an Information Packet (i.e., contiguous or distributed). These are implementation matters beyond the scope of GPP.

C.3.4 SPI_SEQUENCE.confirmation

This function provides an appropriate response to a SPI_SEQUENCE.request indicating the success or failure of the request.

C.3.4.1 Semantics

Figure 32 shows the semantics of this confirmation.

```
SPI_SEQUENCE.confirmation (
                                    Sequence_Tag,
                                    Transmission_Status,
                                    Reject_Reason )
```

Figure 32 - SPI_SEQUENCE.confirmation semantics

C.3.4.1 When generated

This function is generated upon the completion of an attempt to transmit the Information Packet.

C.3.4.2 Effect of receipt

GPP services uses this confirmation to prepare to send the next Information Packet for the same task through the same port or to prepare a request for a different task.

NOTE — GPP permits a negotiated agreement of the number of Information Packets that may be outstanding in each direction between two GPP devices. This action in no way indicates any relationship between these actions, except that the resources for the Information Packet are in a different state or retransmission may be needed.

C.3.4.3 SPI managed parameters

The Sequence_Tag parameter provides a unique identifier of the Information Packet for the service delivery interface. The Transmission_Status parameter shall indicate status as one of the following:

- Successful - Sequence delivered completely to Recipient
- Unsuccessful - Sequence was not delivered completely due to abort or transfer error
- Stopped_by_Recipient - Recipient stopped receipt of the Information Packet
- Rejected_Request - The Information Packet was not sent due to the specified Reject_Reason
- Rejected_by_Destination - Reject indication received from destination

When the Transmission_Status parameter value is Rejected_Request or Rejected_by_Destination the Reject_Reason parameter indicates a reason code.

C.4 GPP implemented SPI bus phases

GPP uses the following phases from the set defined by the SPI:

- BUS FREE phase
- ARBITRATION phase
- SELECTION phase
- DATA OUT phase

All GPP devices shall implement the phases named above. The default mode of information transfer shall be asynchronous transfer mode, at maximum DATA BUS width for the parallel interface.

NOTE — Full width transfers result from disallowing intermix of GPP devices with different DATA BUS widths (see figure 27).

For a single connection, Information Packets for one or more Tasks can be transferred unidirectionally. All Information Packet transfers shall be from the selecting device to the selected device.

Each GPP device establishes a physical path between a seiding port and a receiving port using the SELECTION phase. Information Packet transfers are transferred across the SCSI bus, uninterpreted, to the receiving GPP services for later interpretation. Figure 33 identifies the phase transitions permitted when using the SPI bus for GPP.

C.4.1 SPI BUS FREE phase

The BUS FREE phase indicates that no SCSI device is actively using the bus and that it is available. Sometimes, a GPP port reverts to BUS FREE phase to indicate an error condition that it has no other way to report. This is called as an unexpected disconnect event. GPP devices shall detect the BUS FREE phase according to the requirements specified in SPI.

C.4.2 SPI ARBITRATION phase

The ARBITRATION phase allows one SCSI device to gain control of the SCSI bus so that it can start or resume a Task.

The procedure for an GPP device to obtain control of a SPI is the same as specified in SIP.

C.4.3 SPI SELECTION phase

The SELECTION phase allows:

1 an Initiator to select a Target to make a connection for a new Task.

2 an Initiator to reconnect with a Target after a disconnect.

3 a Target to reconnect with an Initiator after a disconnect.

The procedure to perform the SELECTION phase is the same as in SIP except that the SELECTION phase shall be performed with the ATN signal not asserted.

C.4.4 SPI SELECTION time-out procedure

The SELECTION time-out procedure is the same as required by SIP.

C.4.4.1 SPI asynchronous information transfer (mandatory)

Asynchronous information transfer shall be implemented as described in SPI. It shall be used to perform a synchronous data transfer negotiation (SDTR) using the appropriate message in an Information Packet. See the SYNCHRONOUS DATA TRANSFER REQUEST message in C.2.3.

C.4.4.2 SPI synchronous data transfer (mandatory)

Synchronous data transfer shall be implemented for use during the DATA OUT information transfer phase, except as noted above to negotiate synchronous data transfer mode. Synchronous data transfer shall be used in the DATA OUT phase after a successful synchronous data transfer agreement has been established. The agreement specifies the REQ/ACK offset and the minimum transfer period. The default agreement is asynchronous data transfer. Synchronous data transfer shall be implemented as specified in SPI.

C.4.4.3 SPI wide data transfer (mandatory)

Wide data transfer is mandatory when the DATA BUS width exceeds 8 bits, excluding parity. For wide data busses (i.e., 2-bytes or 4-bytes wide), the service delivery interface shall use a full implemented data transfer width for each REQ/ACK handshake during a DATA OUT information transfer phase. No wide data transfer negotiation is required since connection of data busses of unequal widths shall not be supported. The default agreement is the maximum width of the implemented data bus of each GPP service delivery interface and shall not be changed. Wide data transfers shall be implemented as specified in SPI.

NOTE — If an intermix of bus widths exists on one parallel bus, only GPP devices of the same implemented width can communicate. The assumption is that GPP is used with P-cable or wider implementations and not with the A cable. If a GPP device implements only the A cable, the protocol works correctly with all other one-byte wide GPP devices on the same SPI.

C.4.5 SPI DATA OUT information transfer phase

The DATA OUT phase is the only SPI information transfer phase used with GPP. The phase is used to transfer Information Packets on the DATA BUS.

The C/D, I/O, and MSG signals used to identify the physical information transfer phase are identical to the DATA OUT phase defined in SPI. The port in receiving mode controls these three signals. The port in sending mode can request a continuation of the DATA OUT phase for another Information Packet by asserting the ATN signal during the present DATA OUT phase. The receiving port can disconnect by releasing the C/D, I/O, MSG, and BSY signals and going to the BUS FREE phase as specified in SPI.

Since performing the SELECTION phase without ATN asserted is the signal for using GPP, the first Information Packet transfer of each connection shall not have the ATN signal asserted until after the first complete REQ/ACK handshake of the DATA OUT information transfer phase is completed.

The DATA OUT information transfer phase uses one or more REQ/ACK handshakes to control the information transfer. Each REQ/ACK handshake allows the transfer of one full width bus of information.
The number of REQ/ACK handshakes required is determined by the implemented bus width and the Information Packet length field. Each Information Packet is a multiple of four bytes long so that any implemented SPI bus width (i.e., 1, 2, or 4 bytes wide) can be accommodated without an odd bus width transfer condition.

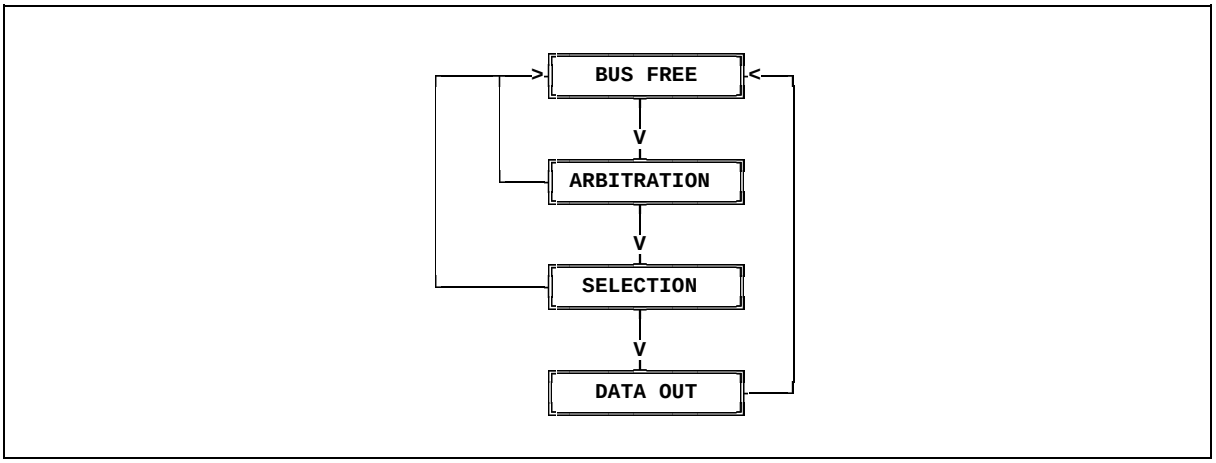


Figure 33 - GPP use of SPI phases

C.4.6 GPP versus SIP operating mode

The GPP operating mode and the SIP operating mode for information transfer are mutually exclusive during a connection.

- It shall be an error to attempt a phase for another operating mode within a GPP connection or a later connection for the same Task.

- If the first information transfer phase of an initial connection is a DATA OUT phase, the receiving port is indicating its intent to operate in GPP mode.

- If the first information transfer phase of an initial connection is not a DATA OUT, then the receiving port is indicating its intent to operate in SIP mode.

Since ports in receiving mode control the selection of information transfer phases, a receiving device shall begin a procedure at power-on or after a reset event, that declares its intent to operate in GPP mode.

- The port selects a GPP address with the ATN signal deasserted.

- The first attempt at information transfer after power-on reset or after processing the reset condition, shall contain a request to negotiate synchronous data transfer. The initiating port expects the selected device to use a DATA OUT information transfer phase.

  The selected GPP or SIP device shall respond with one of the following:

1 If the selected port does not implement Target role, it does not respond to selection (i.e., a selection timeout occurs), and the GPP device discovers that it has selected a SIP device that does not implement Target role, that the SCSI device is powered off, or the SCSI address is unused at the present time.

2 If the selected device implements Target mode and responds to the SELECTION phase without ATN asserted by selecting the DATA OUT information transfer phase, the two GPP devices transfer the Information Packet.

3 If the selected SCSI device only operates in SIP operating mode, implements Target role, and responds to selection without ATN asserted with any information transfer phase other than DATA OUT, the selected port is indicating that it is not a SCSI-3 device and the selecting GPP device shall assert ATN on the first assertion of REQ for any information transfer phase other than DATA OUT and shall not assert the ACK signal. The selected SCSI device shall disconnect from the SPI bus. Both the selecting and selected SCSI devices determine the operating mode of the other and respond accordingly in the future if communication is required, if any.

   If the GPP device decides to switch to SIP operating mode it shall respond to subsequent selections for that SCSI device address using the SIP. If the SCSI device implements only GPP mode, the SCSI device shall not attempt to use GPP mode with that SCSI device address until after a power-on reset or a reset event has been processed.

C.5 SPI events

SPI defines three asynchronous events:

1 attention event;

2 reset event.

3 disconnect event;

These events cause the GPP device to perform certain actions and can alter the phase sequence.

C.5.1 SPI attention event

During a connection, the attention event allows a port in sending mode to inform a port in receiving mode that it has an Information Packet to send. The receiving port gets this Information Packet by continuing the DATA OUT phase. The port in sending mode generates the attention event by asserting ATN as follows:

- not during the BUS FREE or ARBITRATION phases.

- before the ACK for the first byte of any SIP information transfer phase other than DATA OUT to indicate that the Initiator does not operate in SIP mode (e.g., SELECTION). The Initiator may then perform one of the following:

  - not assert ACK which shall cause the Target to time out on the phase, abort the task, and go to the BUS FREE phase (i.e., unexpected disconnect).

  - optionally, respond with ACK, continue to respond with ACK for additional assertions of REQ with 00H byte values and good parity during the same phase, and when the Target switches to the MESSAGE OUT phase, send a SIP ABORT message to the Target. This provides for compatibility with SPI devices. See the SIP standard for this mode of operation since it is outside the scope of GPP.

- during a DATA OUT information transfer phase other than during the first REQ/ACK handshake to indicate that it has an additional Information Packet to send during the same connection. The sending mode port shall assert the ATN signal at least two deskew delays before negating the ACK signal for the REQ/ACK handshake in the DATA OUT phase for the attention condition to be honored before receiving mode port performs the BUS FREE sequence.

A port in receiving mode shall respond to the attention event during a DATA OUT phase as follows:

1 The port in receiving mode shall not disconnect, except for an unexpected disconnect, before responding to the attention event.

2 The port in receiving mode shall complete the current Information Packet transfer, set up for the next Information Packet transfer, and continue the DATA OUT phase.

C.5.2 SPI reset event

The reset event is used to clear all GPP devices from the bus immediately . This event shall take precedence over all other phases and events. Any GPP or SIP device may create the reset event by asserting the RST signal for a minimum of a reset hold time. During the reset event, the state of all bus signals other than the RST signal is not defined.

All GPP devices shall release all bus signals (except the RST signal) within a bus clear delay of the transition of the RST signal to true for the specified minimum time. The BUS FREE phase shall follow the reset event. The reset event has the same effect as defined in SIP.

GPP devices shall implement the soft reset alternative in response to the reset event.

C.5.3 SPI disconnect event

If a sending mode port detects a disconnect by the receiving mode port at any other time than for the conditions below, the receiving mode port is indicating the unexpected disconnect event. The receiving mode port may perform a disconnect independent of the state of the ATN signal. If an Information Packet transfer is in process, the sending mode port shall manage this event as an unsuccessful Information Packet transfer condition.

GPP ports in sending mode normally expect the BUS FREE phase to begin after one of the following occurs:

1 after a reset event is detected.

2 after the transfer of one or more complete Information Packets in the DATA OUT phase.

3 after transfer of 16 or fewer bytes of an Information Packet using the DATA OUT phase. The port in receiving mode is indicating a busy condition. The Initiator may attempt the transfer at a later time.

The busy indication may indicate no buffer is available to accept the Information Packet or no buffer is available which is large enough to receive the complete Information Packet at the time.

4 after an unsuccessful selection.

5 after selection of a SCSI-1 or SCSI-2 device (i.e., selection without ATN asserted). The sending mode port may then decide to operate in SIP mode with that SCSI address. If so, all subsequent communication follows the SIP standard.

All other occurrences of a BUS FREE phase cause the sending port to report an Unexpected Disconnect Event.

ANNEX D
(informative)

SPI information transfer for GPP

For GPP, the DATA OUT phase is used for all Information Packet transfers across a SCSI-3 Parallel Interface. Interpretation of the data in the Information Packet is not be permitted using byte-by-byte processing of the Information Packet on the interface, except the Information Packet length field.

For SPI, the initial Information Packet transfers are in asynchronous data transfer mode. GPP devices do not have an interlock mechanism to identify individual bytes for protocol or parity errors as they are transferred. All interface recovery is at the Information Packet boundary. Only the actual transfer time may be slower than in synchronous data transfer mode. Parity errors are handled by retransmitting the Information Packet.

The information transfer phases defined for the SCSI-3 Interlocked Protocol (SIP) may be used as an alternate mode of operation for GPP devices operating a parallel bus. After detecting a SIP device on parallel bus, a GPP device may change its definition to the SIP operating mode for that device. If such a change occurs, the SIP standard controls the protocol between the SIP operating mode level devices.

NOTE — Single path mode in GPP is equivalent to normal operating mode in SCSI-2 and SIP. Also, the assign and unassign functions of GPP replace the reserve and release functions in SCSI-2 and SIP.

D.1 Operation notes

Since GPP devices support selecting another device and being selected by another device, Information Packet transfer permits a common transfer protocol in all ports. Although SPI is a half-duplex bus, GPP treats the bus as a blocked full-duplex environment. That is, when the bus is busy, a GPP device cannot select another device. This symmetry of operation permits a common implementation of the GPP protocol for Initiators and Targets.

The overall simplification of phase use can lead to internal simplifications of parallel bus devices as well. Energy, and money, spent on the complex interaction between phases can be applied to improving the throughput of the system on both ends.

The paradigm used for GPP differs considerably from SIP. SIP assumes that an Initiator is instantaneously capable of supporting Target requests once a Task begins. GPP permits overlapped transfer of information for the same Task with processing of earlier information for the same task. This eliminates much of the real-time aspects of SIP and replaces it with a fully buffered information exchange protocol.

D.2 Application examples

Since SPI is the base form which other SCSI-3 protocols and interfaces branched, describing some applications may stimulate extensions to using the original SPI and may stimulate migration to new, faster, more reliable protocols and interfaces.

D.2.1 Migration to new interfaces

Once GPP is implemented for SIP, other interfaces, as specified in other annexes in GPP, become candidate migration paths for either higher performance or more flexible or reliable interconnect environments (distance and number of ports).

D.2.2 High availability environments

GPP also permits multipath operation within the context of SIP. A classic configuration is a dual ported Initiator and a dual ported Target. Each pair of ports in the respective devices is connected by a single SPI. The result is a two-bus system.

- In SPI, there is no interaction between these two busses, except for some reset management. Should a path fail, all Tasks pending in a Target must be aborted. The alternate path cannot be used to continue the Tasks. The Initiator must perform a recovery action for each Task. This may either be as simple as reissuing the command or it may be a very complicated procedure.

- With GPP, the two logical elements can discover the other by an exchange of messages that identifies the end points (i.e., the logical elements) on the two busses. This process identifies two paths between the logical elements. With this information, the Initiator can group the two paths into a path group.

  The result of this process is to permit any Task to start on either pair and continue on either pair for performance enhancement or in the presence of a path failure. In either case, the entire recovery process above is eliminated. A double failure could cause the procedure above to be invoked, but all communication is halted in the presence of two errors in either case.

D.2.3 Processor-type clusters

A natural extension of GPP, is that clusters of processor-type Logical Units can be easily interconnected. Distributed processing environments are a natural outgrowth of the GPP paradigm. With the wide selection of service delivery interfaces, almost any arbitrary collection of logical elements can form a cluster of processor-type Logical Units. Asymmetric connections can be used to form subgroups of processor devices.

D.2.4 Third-party operations

Since GPP operations are not restricted to a single bus, third party operations are possible. A copy manager function sitting at the collection point of several service delivery interfaces could manage copies among several busses (and other interface types as well). The copy manager can also act as an interface-type bridge during the COPY command operations.

Easing the restriction for COPY operations to be limited to a single SCSI-3 bus open up considerable opportunity to offload data archive tasks. The management still belongs to the Initiators, but the data transfer operation can be delegated to a COPY manager with access to several busses. A source or destination is identified by the WWN(t or i) || LUN. A copy manager can process a command if it can reach both the source and destination on any service delivery interface it can access.

Annex E
(normative)

Internet Protocol usage for GPP

This clause specifies using an Internet as a service delivery subsystem for GPP. This annex addresses a service delivery subsystem that specifies the logical and physical requirements for interconnection using the Internet Protocol (IP) suite.

For details on the Transmission Control Protocol/Internet Protocol suite (TCP/IP) suite and its supported service delivery subsystems, see the latest IETF documents, called Request for Comments or RFCs. The corresponding physical interface standards are identified in those documents. The TCP/IP developers have one RFC document that identifies the latest versions of the Internet protocols. See that RFC to determine the proper level of the other RFCs.

NOTE — The Internet does not have official approved standards the way ANSI does. All RFCs are widely reviewed and managed by a central administrative group, called the Internet Activities Board or IAB. Every 3 months, the IAB releases an RFC with the latest RFCs for the protocols identified. RFCs are available electronically from the host NIC.DDN.MIL.

- The File transfer protocol (FTP) is used to GET the file. The file names are of the form <rfc>rfcN.txt, where N is the number of the RFC (e.g., <rfc>rfc793.txt for one of the TCP RFCs).

- Through electronic mail you can send a message to service@nic.ddn.mil. The SUBJECT field specifies the RFC you want (e.g., RFC 793). You can also fill in the work "help" to get a general information document.

- Paper copies are available by calling 1-800-235-3155.

- The file <rfc>rfc-index.txt contains an index of the latest RFCs and those at the latest level.

Popular physical transport layers for exchanging Internet Protocol information are Token Ring, Token Bus, Ethernet, FDDI, and RS-232 (e.g., over telephone lines). GPP places no new requirements on those service delivery subsystems or any others not mentioned but supported by the TCP/IP suite.

Although the IP documents are not ANSI or ISO approved standards, conformance with documents produced in the IP community is high, probably higher than for ANSI or ISO standards like SCSI-2. Therefore, these documents may be used with a high expectation that attached systems and devices will conform to the specified protocols. GPP only requires that the systems implementing GPP over TCP/IP have the required protocols and TCP. Most TCP/IP packages contain many more functions than required by GPP.

This annex and annex F specify the use of the Internet Protocol as a service delivery subsystem for GPP. A GPP logical system may be constructed using any combination of the service deliver subsystems specified for GPP.
The level of service required to support GPP is as an application above the Transmission Control Protocol/Internet Protocol (TCP/IP) suite. The basic mechanisms of IP support identifying a logical port or socket above TCP/IP that sends and receives GPP Information Packets.

E.1 IP specific messages

There are no IP specific messages defined for GPP.

E.2 Encapsulation of an information packet

Encapsulation of an Information Packet for transmission over IP requires that the total packet length not exceed a length of 65 536 bytes minus the IP and TCP packet header overhead. Since some hosts may use optional fields allowed by IP, this limit is 65 280 bytes. The total IP packet length is 65 535 bytes including the IP header and TCP header information. This Information Packet length restriction permits properly constructing Information Packets which can be transmitted by IP. For those service delivery options that cannot send an Information Packet as a single entity, the TCP/IP protocol suite is responsible to segment the Information Packet into acceptable transfer units and reassemble the Information Packet at the destination.

The process of filling in the IP header fields is well defined in the appropriate TCP and IP documents and will not be duplicated in GPP. These interfaces and their primitives are fully described in the Internet RFCs. Because of the nature of SCSI Tasks, a reliable data delivery service is required. GPP requires that the use of TCP/IP rather than UDP/IP.

E.3 TCP/IP services

IP provides a connectionless, unreliable data delivery system with out of order delivery of data possible, or data loss. TCP provides a connection oriented reliable stream data delivery system that provides the destination a view of the data stream between two hosts as if they were connected by a single link. At the receiving host, TCP is responsible to reorder out of order segments and to recover lost segments of information from the source.

Only information reassembled correctly at the destination host is presented to the destination application. Therefore, TCP/IP provides a well-ordered transmission of information between source and destination. The stream view of information is maintained by TCP using a SEQUENCE NUMBER field in the TCP header.

The sliding window function of TCP permits streaming information between the two hosts in a full duplex manner. TCP segments and acknowledgements flow in both directions at the same time. This is consistent with the full duplex nature of GPP and does not contribute any interface specific provisions for GPP services. The sliding window provides for flow control between the two hosts as well as some TCP defined mechanisms for congestion management in the network. Congestion management is outside the scope of GPP.

TCP provides a concept of pushing correctly received information to a destination by forcing a TCP segment to be sent when it is shorter than desirable and also pushing the data to the destination host application when the receiving buffer is shorter than desirable. This function is used in GPP to mark the end of Information Packets.

Checksums on the GPP data as well as the TCP and IP headers is supplied by TCP and IP. Therefore, data corruption is not an issue when using TCP/IP.

Certain port numbers in TCP and IP have been reserved for well-known functions. These well-known port numbers shall not be used for GPP operations. However, each host may use the full functions of TCP/IP for any other purpose it has in addition to GPP. These operations may involve the use of these well-known port numbers.

E.4 TCP/IP support of SAM

TCP/IP has well defined link level responses and control processes to assure reliable Information Packet delivery between an Initiator and a Target. TCP/IP specifies the exact manner in which an IP packet shall be transmitted, the normal responses and the abnormal responses. These specified responses shall be used by GPP and no new responses or control requests shall be required. No equivalent vendor unique response types or field values shall be specified.

The strict ordering of the delivery of information between two hosts permits ordered and head of queue operations as well as simple queue management operations to behave as specified in SAM.

The strict ordering also causes Task Management functions to arrive in the correct order as sent by the Initiator of a task. That is, the state of a logical unit is approximately the same as would be observed on a SCSI Parallel bus if the same stream of information were transferred in the same order. This provides remote device behavior similar to that of a local device.

Logical responses to the content of an Information Packet are generated by the receiving logical element, where applicable, as new Information Packets. Expedited data and other rarely used features available in TCP/IP shall not be used.

E.4.1 IP header

The IP header length for GPP supports the IP minimum of 5 words or 20 bytes.

The SERVICE TYPE field, which is a processing request field, supports the following values for GPP:

- PRECEDENCE = 0, for normal precedence
- D = 1, for low delay
- T = 1, for high throughput
- R = 1, for high reliability.

IP OPTIONS field support is not required by GPP. If the options field is used, it is used by a layer below GPP services. Therefore, the PADDING field is not necessary unless required to support options use.

E.4.2 TCP header

The TCP header length needed by GPP is the minimum of 5 words or 20 bytes.

The TCP header has many fields that are managed solely by TCP and are of no interest to GPP. However, the CODE BITS field has one interesting property of use to GPP and available for use by all applications programs. The function is called push and uses the PSH subfield of the CODE BITS field. Other fields in the CODE BITS field are used by TCP as appropriate for its protocol management and are not invoked by GPP.

GPP services shall use the push function to mark the end of each Information Packet. This provides an interlock between applications that data has arrived at the application. When TCP acknowledges that the last byte before a push function has been received, GPP services knows that the Information Packet has arrived at the destination application.

The TCP OPTIONS field is not used by GPP. Any use of this field is at a layer below GPP services. The PADDING field is used only if required by the OPTIONS field use. GPP does not use this field.

E.4.3 TCP/IP header overhead

The total overhead added to an Information Packet by TCP/IP is 40 bytes, minimum. GPP services only requires this minimum overhead. If TCP/IP fragments the Information Packet into segments, then the overhead of 40 bytes, minimum, is added to each segment.

Acknowledgement overhead is contained in the 20 byte overhead of the TCP header and does not add any additional overhead. Acknowledgment information may be included with a TCP segment going the opposite direction. TCP does have the right to send acknowledgements with no GPP payload present.

E.5 Service interface from GPP to TCP/IP

The service interface for an IP packet is specified in the appropriate IP documents. GPP is an upper layer protocol to TCP/IP. The service protocol currently specified for TCP/IP shall be used. Invoking these services causes an Information Packet encapsulated in an IP packet to be transmitted to a destination. Because of the potential distances and time involved for transmission, a packet streaming protocol is specified in TCP/IP. This protocol may be used by GPP to provide a performance enhancement. This streaming protocol is consistent with the information packet prefetch operations specified in the body of GPP.

TCP does not specify a precise protocol or interface from the application to TCP. Each platform has its own internal architecture which must be observed. This might be compared to the interface for SCSI host adapter cards. Each vendor has the right to specify an interface. However, over time, there has been some convergence on two or three major interfaces. GPP services is the component of GPP responsible to match the TCP interface and not the application client or the logical unit.

Since TCP is a connection-oriented service there is a series of steps to set up, use and tear down a connection. The general sequence of events is outlined below:

- Establish the connection - This uses a TCP CODE BITS field called SYN (pronounced "sin") to synchronize the two hosts. When this protocol completes successfully, a connection is established and GPP Information Packets may be transmitted.

- Established connection - At this time GPP information packet transfers, error recovery of lost segments, etc., occurs. If either host needs to break the connection, that host starts a close process.

- Closing the connection - During this process, one of the hosts requests to close the connection by sending a TCP segment with the CODE BITS field FIN set. This begins a sequence of events that breaks the connection.

When the connection is closed, no further GPP Information Packets can be transferred until another connection is established. It is not considered a GPP error for TCP or GPP services to close the connection. A new connection may be established to continue operations as required.

One option of TCP permits a connection to be reset rather than closed. When the RST subfield of the CODE BITS field is set, a connection is immediately terminated and each application is notified. If a partial Information Packet has been received at the application, it shall be discarded. All buffers in TCP at both hosts are released. This event only breaks the connection and does not cause either logical element to abort any tasks. That is, a TCP connection reset has no effect on logical operations between the GPP devices.

The exact method of invoking these services depends on the host platform and the TCP/IP implementation available. See the system documentation for precise details.

Annex F
(informative)

Internet Protocol information transfer for GPP

The Internet (IP) system is by far the largest internetwork in the world. A basic premise of the Internet is that any independent network can be added to any other network and the various users can communicate. Therefore, Internet is a network of networks. With GPP mapped to this system, any single component network becomes a candidate for implementing GPP. Internet is a set or suite of protocols that specifies how an application is to use the Internet to communicate with another application within its own host or with a remote host.

Outside the host, the Internet has additional protocols for mapping one network into another. This level of protocols is beyond the scope of GPP and is concerned with the term gateways.

GPP concentrates on the interface between the GPP services (i.e., the application) and the TCP/IP portion of the protocol suite. This focus means that the logical unit and application client of SAM are not concerned with the details of the process of transmission. GPP retains the separation of these components from the service delivery subsystem. This limits the focus for definition and implementation to a much smaller region of the complete Internet set of protocols.

GPP focuses on the Transmission Control Protocol/Internet Protocol (TCP/IP) since it provides the level of data delivery reliability expected in SCSI. Because of the nature of the TCP/IP, a logical connection is established between the two cooperating nodes. After the logical connection is established, each node may expect Information Packets from the other. If a logical connection is not established, the Information Packets are not sent; another connection attempt may be made. This login process is specified in the appropriate IP documents. The receiving port is not aware, other than that a logical connection exists, that an Information Packet is about to be received. The sending port may not be aware of the present state of the destination port.

GPP assumes the delivery of complete Information Packets and recovers on that boundary when an error occurs. However, TCP/IP recovers on the segment boundary (i.e., a portion of an Information Packet). TCP/IP is unaware of the boundary between Information Packets since TCP/IP manages a stream of bytes. A special function of TCP/IP is used to force delivery of the end of each Information Packet out of the source port and to the application at the destination port. Other than this, TCP/IP has no concept of the application data structure.

This implies that the source and destination applications are synchronized on the data streams (i.e., one in each direction) between them. The structure of GPP Information Packets provides the destination application with a length indication for each Information Packet. The destination application uses this field to mark the ends of Information Packets.

F.1 Error Management

Error free Information Packet delivery is not guaranteed at the time of transmission. The TCP/IP processes provide for positive responses from the opposite end of a logical connection. TCP/IP is responsible to generate appropriate acknowledgements for IP packets according to the TCP/IP processes. A partially received Information Packet shall not be processed by the GPP application and shall be discarded by the receiving GPP application. GPP services is the component responsible to provide only complete Information Packets to the receiving application.

There is no data dependency between the IP and TCP headers and the GPP Information Packet content. The TCP/IP headers are not to be processed by a receiving GPP application. The only level of interaction is isolated to GPP services. Only the net payload, an Information Packet, is to be processed by a GPP application.

The path between two GPP devices is considered to be a full-duplex, fully operational bi-directional transmission path. The actual mechanisms used by the TCP/IP services to manage this method of operation is beyond the scope of GPP. The TCP logical connection between two nodes may be used for more than one Task. A separate connection need not be established for each Task. GPP isolates the logical content from the service delivery subsystem.

This independence from the service delivery subsystem permits GPP to use cooperating sets of ports, two or more pairs to accomplish a single Task. Multiple port pairs, when available, may be used to increase the usable bandwidth between the nodes as well as to provide a highly available system environment.

GPP services is responsible to start and stop connections and manage the flow of complete Information Packets between application client and logical unit. If a connection is terminated for any reason, the receiving GPP services discards any partial Information Packet(s). The source GPP services is also notified of any termination of a connection. If any Information Packets have not been fully acknowledged, they are made available for retransmission when a connection is reestablished. Of course, if a timeout occurs between the source logical element and the source GPP services, this is reported to the logical element in the normal manner specified in clause 8.

F.2 The value of the sliding window

A key feature of TCP/IP is to attempt to maintain a stream of information flowing between the two GPP devices consistent with the Task load between them. The sliding window provides a means to have multiple TCP segments in flight across an internet with the expectation that an acknowledgement will follow. Since TCP/IP connections are full duplex, the deferred acknowledgement mechanism of TCP segments permits considerable information to be in flight in both directions at any one time.

The time to delivery and acknowledge a TCP segment can be considerable. On a single network the time can be relatively short. However, across the full internet, an acknowledgement can take several seconds. The time to deliver is also affected by the congestion of the gateways in the internet. Sufficiently long timers in both the application client and in TCP/IP are required to permit information to flow in both directions.

The structure of GPP, unlike a native protocol, permits the minimum interaction between application client and logical unit to complete necessary work. If an application client sends only a WRITE-type command and no data along with it, it could take 8 to 12 seconds to perform the command over a long distance or in the presence of congestion. With the sliding window, the process for a native protocol emulation might go as follows:

- Application client sends a WRITE-type command with PSH = 1b (2 seconds)
- Logical Unit interprets the command
- Logical Unit sends a request for data with PSH = 1b (2 seconds)
- Application client receives request and interprets it
- Application client sends write data with PSH = 1b (2 seconds)
- Logical Unit processes the data and prepares status
- Logical Unit sends status and completion indication with PSH = 1b (2 seconds)
- TCP/IP sends a final acknowledgement for the status (2 seconds)

- Total time 10+ seconds.

With GPP, the interaction is altered as shown below:

- Application constructs and sends one Information Packet with the WRITE-type Command and the data with PSH = 1b (2 seconds)
- Logical Unit interprets the command, processes the data, and prepares status
- Logical unit sends status and completion indication with PSH = 1b (2 seconds)
- TCP/IP sends a final acknowledgement for the status (2 seconds)

- Total time 6+ seconds.

The wait for the final acknowledgement at the logical unit is to guarantee that the status arrives correctly. The Task ends at the application client with receipt of the status, but only after the final acknowledgement at the logical unit.

With a smaller or less congested system, the times will be smaller in either example but proportional in any case. In either example, new Tasks may be started and overlapped up to the queuing depth of the logical unit. However, the difference in latency for a single Task with the sliding window is obvious from these examples and requires attention in the protocol to reduce it as much as possible.

If we eliminate the sliding window, each TCP segment must be acknowledged before the next segment can be sent. In the examples above, assume the data is 8 kilobytes and the TCP segment size is at the minimum (under about 600 bytes). Assume about 512 bytes of data per TCP segment to keep it simple.

- The first example requires a total of 19 TCP segments be transmitted with the corresponding acknowledgements:

  command - 1
  data request - 1
  data - 16
  status - 1.

- The second example requires a total of 17 TCP segments be transmitted also with the corresponding acknowledgements

  command plus data = 16
  status = 1.

Without the sliding window used in the earlier examples, the time would be increased by 60 seconds in both examples. The round trip time for TCP segment plus acknowledgement is about 4 seconds. There are 15 extra segments for the data for an additional 60 seconds in each example. Without the sliding window, the difference between 70 seconds and 66 seconds seems inconsequential.

F.3 Performance

There are good and not-so-good implementations of TCP/IP as there are with other interfaces. However, some of the best TCP/IP implementations have seen sustained data rates of 8 Mb/s on a 10 Mb/s local Ethernet. More general implementations may achieve 200-400 kilobytes per second at some distance.

TCP/IP as a service delivery subsystem for GPP is not meant for high performance requirements. It is meant for connectivity. Often connectivity is more important than speed. However, there are some very fast TCP/IP systems in existence that would rival a local SCSI parallel bus.

Annex G
(normative)

Heterogeneous InterConnect usage for GPP

This annex describes the physical characteristics of Heterogeneous InterConnect (HIC) standard (IEEE P1355) as it applies as a service delivery subsystem for use with GPP. HIC specifies the physical requirements for a low cost, low latency, scalable serial interconnect for parallel system construction. See Clause 2 for the complete titles and document numbers for this standards.

This annex and annex H specify the use of HIC as a transport layer for GPP. A logical system may be constructed using any of the service delivery subsystems specified in GPP or a combination of two or more.

G.1 HIC definitions for GPP

G.1.1 control character: an encoded set of bits transmitted betwen two devices to signal control information to a device.

G.1.2 data character: an encoded byte suitable for transmission between two devices.

G.1.3 Destination: a field at the beginning of a packet that is used to route a packet to its destination terminal Node. In the SCSI-GPP mapping, it is composed of one or more destination IDs.

G.1.4 device: the name of any HIC element that attaches to a HIC link (e.g., a computer or a switch).

G.1.5 End-Of-Packet Marker: a control character that marks the end of a packet.

G.1.6 fibre: an optical strand or a small set of electrical wire used in a serial manner to transmit information.

G.1.7 image: a function within a Node that is assigned a unique destination ID and implements a GPP device functionality. Two or more images may be present in the same Node, each being assigned a unique destination ID.

G.1.8 link: a bidirectional means of communicating point-to-point between devices.

G.1.9 Node: a logical element; a GPP device.

G.1.10 packet: a HIC unit of transmission consisting of a Destination element, a Payload (0-n data characters), and an End-of-Packet Marker.

G.1.11 parity: a field in a character used for parity checking.

G.1.12 path: the set of links used in the transmission of a packet from a source terminal node to a destination terminal Node.

G.1.13 Port: an external access point for a Node to a HIC link; a GPP port.

G.1.14 transmission character: an data character or control character transmitted between two devices.

G.2 HIC specific messages

There are no HIC specific messages defined for GPP.

G.3 HIC physical description for GPP

Each GPP device is called a *Node* in HIC terminology. Each Node has at least one *Port*. Each Port is connected by a bi-directional HIC connection. The HIC standard for a HIC network defines implementations using different types of conductors, data encoding schemes, and speeds (e.g., optical, copper serial, etc.).

All HIC link operations are conducted through the transmission of *control characters* and *data characters* in the form of *packets* on links. The number of bits in the bit stream for a control character may be different than for a data character. A packet is made up of a *Destination*, the *Payload*, and an *End-of-Packet Marker*.

A GPP device sending a packet is known as a *source terminal Node* and the GPP device to whom the packet is addresses is known as a *destination terminal Node*. Generically, they are called *terminal Nodes* when no distinction is to be made. Several different links of any combination may be traversed by a packet as it moves from a source terminal Node to a destination terminal Node through a HIC network. The set of links used for transmitting a packet is called a *path*.

Inside each Port is a *Source* that provides packets to be sent to some destination. These packets are properly converted by HIC to a serial bit stream and transmitted to a destination terminal node. At the destination the bit stream is converted back to transmission characters. Data characters are converted back to bytes and sent to a *Sink*.

For GPP, the Source and Sink represent a GPP device in either Initiator role or Target role. Since HIC has bi-directional links, each GPP device acts as both a Source and a Sink.

A set of packets that performs a coordinated action between two terminal Nodes is called a *transaction*. HIC is not aware of the nature or structure of a transaction, except that it is asked to send packets and packets are received and directed to the correct destination internal to the Node. For SCSI-3, a transaction is equivalent to a Task.

With HIC, each GPP device shall assume that it is connected to a packet-switched network. In HIC, a network is any sequence of devices connected by links that join source terminal Nodes to destination terminal Nodes. A single link connecting two devices also fits this definition (i.e., there is just a source and a destination device and a single link between them). When using a HIC port, a GPP device may be physically connected to another GPP device using:

- a point-to-point topology by a single link(e.g., an Initiator and a single Target with Logical Units)

- a switched point-to-point topology with two or more links in the path between terminal Nodes (e.g., an Initiator connected to a switching device which is then attached to several Targets).

G.3.1 Destination ID assignment and Destinations

Destination IDs are assigned to terminal nodes and not to the ports that attach the Nodes to HIC links. A Node may have many destination IDs. The destination IDs are statically assigned to the Nodes and the network is conditioned to route packets to the Nodes.

In some configurations where a Node has two or more ports, the packets for a single destination ID may arrive on any defined port since the destination ID identifies the Node and not the port used to deliver a packet to the Node.

Conversely, a single port may receive packets for several different destination IDs and that port may be the only port through which some destination ID may be delivered.

A source terminal node must assign the *Destination* for each packet. The Destination uniquely identifies the Node (not the port). In simple systems, this is an encoded one byte identifier for an address space of 2-256 devices. In larger systems, additional destination IDs are prepended to the final destination ID to specify a unique Destination.

The Destination is used for identification of destination terminal Nodes and also, implicitly, to route a packet through certain intermediate devices in the network. These intermediate devices are normally switching devices that select some output port based on the first destination ID found in the Destination.

HIC uses a *destination routing*. A network willl deliver each packet somewhere. The same header is used to determine the output port of a switch. The switches are configured so that two packets with the same header that enter a HIC network at two different points are delivered to the same destination.

G.3.2 No acknowledgement of packet delivery

In HIC, there is no direct link level acknowledgement by a HIC device that a packet arrived at its destination. However, the reliability of a HIC system makes corruption or loss of part of a packet a very rare event (approximately a 10E-15 error rate). Most Initiators currently have process timers on commands. This function can be used to detect the loss of or corruption of a packet in a Task.

Except for some Task Management functions, each Task in GPP requires that at least one packet be returned from the Target to the Initiator. This provides a high level interlock on error detection consistent with the low error rate of the physical system.

The TASK WAITING message provides a method to inform another SCSI device that it considers the elapsed time for a command or Task to be excessive, but not yet aborted. The definition of excessive in this instance is an implementation specific value and is not specified. Also, a GPP device is not required to respond to a TASK WAITING message in any manner.

G.3.3 Order of delivery of packet contents

Information in HIC flows in units called packets. The contents of a packet arrives at a destination terminal nodein the order it was transmitted.

G.3.4 Order of delivery of independent packets

Although the contents of a single packet arrive at the destination terminal node in order, two independent packets destined for the same destination terminal node from the same source terminal node may arrive out of order. It is possible to permanently configure a HIC network to provide in-order delivery of all packets between two terminal nodes. However, this might not provide the fastest network for delivering packets and many packets have no order relationship between them that is not already managable with the internal structure GPP imposes on each task..

Within a Task, GPP provides internal sequence control information so that the destination terminal node can, if necessary, rearrange packets for the correct order of processing. For GPP, a destination terminal node may discard valid packets for any reason up to the negotiated retransmission depth to attempt to get packets within a Task to arrive in order.

However, a GPP device is required to correctly process packets within a Task that are received out of order. Discarding packets as a way of attempting in-order arrival of packets is not encouraged since it increases Information Packet traffic and reduces useful bandwidth on the network, but it is an available technique for attempting to handle out of order receipt of packets at a destination terminal node. Intermediate devices are not aware of, nor need to be aware of, the logical relationship between packets in a transaction.

G.3.5 Relationship of a packet to an Information Packet

For GPP, an Information Packet shall be transmitted as a single packet when using HIC. This simple one-to-one mapping makes it easy to transform the packet to another physical interface without a logical process of collecting two or more packets to form a new unit or attempting to split a single packet into two or more smaller packets.
HIC has no upper limit on packet length. GPP limits the packet size to a payload of 65536 bytes. For a HIC network, extremely long packets may degrade system performance if a long packet gets stalled in its transmission across the network to a destination terminal node. However, no HIC failure results from having a packet stalled in the network.

Implementors are encouraged to, but not required to, vary the Information Packet sizes as the load on a network increases. The determination of a load change is implementation specific and beyond the scope of GPP. GPP does provide tools, in the form of the messages, to dynamically negotiate the maximum Information Packet length between terminal nodes. This negotiation between two GPP devices may be asymmetric for the maximum Information Packet length to better balance resources in both GPP devices.

For example, when writing data to a disk drive, the packet length outbound may be considerably longer than the packet length inbound where usually only Status is presented. For long periods of reading, the reverse arrangement of Information Packet lengths may be appropriate.

The negotiation may be initiated by either GPP device at any time. The result of the negotiation is persistent, both not permanent, until changed or reset because of a GPP device reset.

G.3.6 Managing the order of execution of commands with HIC

Since the network may alter the order of delivery of packets to a destination terminal node in some HIC networks, some care must be taken in managing a sequence of commands that must be executed in a specific order. Single packets for two independent Tasks sent to the same destination terminal node may arrive out of order at the destination terminal node, except when the entire network is designed to deliver all packets in order, because of the characteristics of the HIC service delivery subsystem. In this case, the use of ORDERED Tagged Tasks to cause ordered execution of two or more Tasks may fail to achieve the desired behavior in a Target (e.g., for a sequential device, a READ command is expected to follow a REWIND command but the commands arrive our of order in independent tasks and get processed as a READ command followed by a REWIND command).

A similar argument may be used for any perceived mis-ordering of HEAD OF QUEUE Tagged Tasks. The tasks are placed into the Task Set of the Logical Unit in the order received at the Logical Unit and not in the order transmitted.

SAM provides an optional function for constructing tasks called linking. Linking permits several commands to be processed in an Initiator specified order by linking them together in the desired order in a single Task. When this feature of SAM is used for a Task, the commands within the task are processed in the order specified by the Initiator, independent of the order of arrival of the individual packets associated with the Task from the Initiator.

Because the natural out of order delivery of packets that may occur with HIC, the optional linking function of Logical Units may be necessary in a logical system to force the correct ordering of a sequence of commands in an efficient manner. Of course, the Initiator can always solve this ordering problem by holding the queue of commands itself and sending each subsequent command in a new Task after the previous single command Task finishes successfully. This

may defeat the performance goals of the system, because of the latency it may introduce. This should be considered a last alternative to processing a sequence of commands in an Initiator specified order.

G.3.7 Simple GPP service delivery subsystem with HIC

The simplest implementation of a SCSI-3 service delivery subsystem using HIC requires that all destination IDs in a network be GPP devices. That is, an entire HIC network consists of only GPP devices. Other networks in same system may implement other functionalities (e.g., ATM functions). This type of implicit, homogeneous system is similar to that found on a SCSI-3 parallel interface (SPI), where all devices present are SCSI devices. In such a network, each destination ID is either a GPP image or it is not present.

The simplest HIC network allows a maximum of 256 destination IDs (GPP devices) with a switching network in the center and a single level Destination structure. The Destination is composed of a single control character identifying a single destination ID (00h-FFh). The network routes each packet to a destination terminal node with that HIC destination address.

In this service delivery subsystem, the GPP addresses in an Information Packet are mapped one-to-one to the byte version of the destination IDs. Obviously, for a more complex network with more than 256 destination IDs, there must be a mapping from the Destinations to a single byte value.

The routing to each terminal node is statically assigned in each HIC system. If the devices are plugable entities, they may be present or not. Consistent with SCSI-3, any destination ID could be associated with any type of GPP device.

Each GPP device has a simple, non-invasive method for determining its own destination address without requiring a manual configuration setup of the GPP device (i.e., no DIP switches or other mechanical mechanism are required). HIC permits a null packet to be transmitted and correctly received (i.e., a packet with no payload) which in some HIC physical interfaces is as short as 14 bits long.

Since there are only 256 possible GPP devices, a simple poll to each of the 256 possible destinations with a null packet directed to each destination ID will cause at one null packet to be routed to the input of the transmitting GPP device. Since there is no acknowledgement of receipt, a GPP device cannot tell, and need not tell, whether a null packet it receives it its own or sent by some other GPP device.

The Destination in any packet, including a null packet, identifies the receiver of the null packet and, thus, provides the destination ID of the polling GPP device. The may poll stop when the first valid packet of any type is received since any valid packet received at the GPP device provides the destination ID of that GPP device.

Further, GPP requires that each GPP device attempt to identify the Targets (for an Initiator) or the Initiators (for a Target) in a logical system. Again, since this is a small address space, INQUIRY commands sent to each possible Logical Unit (using the CAM polling technique) will identify the available GPP devices (at least one other), the Logical Units available in those GPP devices (at least one in each GPP device), and the device class of each of those Logical Units.

In HIC since a destination terminal Node may support more than one destination ID, two or more independent GPP devices may reside in the same physical terminal node. If these multiple destinations are through the same port, the HIC neetwork must be appropriately configured. Since they have independent destination IDs, the Initiators can communicate with a local GPP device in the same manner that it communicates with a remote GPP device. In Fibre Channel, these independent functions in a Node are called images. Each image represents a totally independent GPP device. In GPP, it is assumed that GPP devices only know of each other and only communicate with each other using the SCSI-3 constructs and the service delivery subsystem, in this case, the network provided by HIC.

Since SCSI-3 supports processor type devices as well as other types of devices, GPP may be used to implement interprocess communication with the Processor command set, where each process has a GPP functionality and is

independently identified in the HIC network. GPP provides a single consistent communication system between processors and peripherals and between two processors.

One purpose of the INQUIRY poll required by GPP is to permit a late arriving GPP device to provide an Asynchronous Event Notification of its presence to each Initiator. Conversely, an Initiator can discover the Targets it has available to use. With images as defined above, starting a process in a Node could have the effect of adding a new GPP device to the configuration that has an independent functionality. This provides a self-configuring environment and dynamic configuration change environment for an HIC system.

In SCSI-3, all information flow is given a direction from the point of view of an Initiator. The flow of information in SCSI-3 is outbound from Initiators and inbound from Logical Units. Because of the full-duplex nature of HIC, the actual difference in the direction of information flow can only be determined by examining the content of the Information Packet. The physical transport layer in HIC is symmetrical in all transfer operations.

```
Leaving a Port

        Transaction Level (TL)
                Performs SCSI-3 device model (Initiator or Target roles)
                Results in an exchange of packets with another SCSI-3 device

        Packet Level (PL)
                Converts bytes in a packet to transmission characters

        Exchange (Link) Level (EL)
                Adds control characters

        Character Level (CL)
                Converts transmission characters to a bit stream

        Signal Level (SL)
                Transmits the bit stream


Entering a Port

        The processes are reversed for the inbound fibre
```

Figure 34 - HIC terms

G.4 HIC encapsulation of an Information Packet

The GPP unit of work between an Initiator or a Target is an Information Packet. The equivalent HIC construct is called a packet. The HIC packet delivery protocol provides the building blocks for conducting GPP logical operations. In HIC, GPP logical operations are called transactions.

G.4.1 HIC packet

With HIC a logical element requests that a packet be transmitted to another destination ID. The request is in the form of a packet with a destination ID list sent from a logical element to the HIC packet level (PL). The HIC link management function at this level handles all details concerned with building a Destination from the destination ID list and physically managing transmitting a packet.

G.4.2 Information Packet management

This sub-clause specifies a service interface to isolate GPP services from the HIC LL. When an Information Packet is to be transferred, certain information must be supplied along with the Information Packet contents. The only control information added to a GPP Information Packet (i.e., the Payload) by HIC is the Destination at the front of a packet and the End-Of-Packet Marker.

The GPP services described in an earlier clause provide the mechanism to handle Information Packets to support a Task. This clause provides a service layer to invoke packet handling services at the LL in HIC.

In an operating system, a device driver is usually responsible for translating application requests to I/O requests. At the Target end, each Logical Unit is responsible for translating responses to Information Packets. GPP services provide isolation between the GPP device and the GPP protocol.

To permit interoperability of GPP devices, this level of detail is required to properly construct a protocol service level which operates between the GPP services and the HIC LL functions in each Node.

G.4.3 HIC responses and signals

HIC provides a protocol stack to manage the various stages in packet handling that it defines. HIC specifies the exact manner in which an information unit shall be transmitted and any abnormal responses. These specified responses shall be used for GPP. No additional response types, field values, or vendor unique values shall be specified.

G.5 GPP services to HIC Link Level services

Because this clause bridges between GPP Services and HIC packet management functions, the term *Link Level (LL)* will be used to refer to all HIC packet handling services starting at the *Packet Level (PL)*.

The *Transaction Level (TL)* in HIC is equivalent to GPP services. The LL services are used by a GPP TL to cause an Information Unit transfer to another GPP TL.

The following LL service functions support GPP Information Unit transfers between GPP services and HIC LL:

HIC_PACKET.request

HIC_PACKET.indication

HIC_PACKET.confirmation

G.5.1 HIC_PACKET.request

The HIC_PACKET.request function specifies sufficient information for the transfer of one information packet from a TL to a single peer TL.

All parameters in this function shall provided. If a parameter is to be assigned to LL, the invocation indicates that no value is supplied by the TL.

G.5.1.1 Semantics

The figure below indicates the semantics of this request.

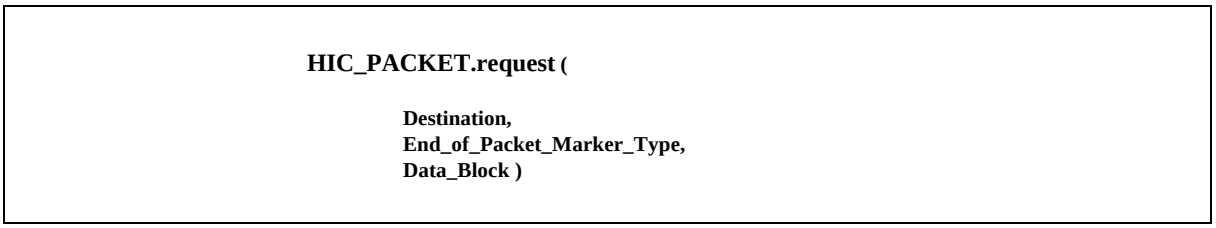


Figure 35 - HIC_PACKET.request semantics

G.5.1.2 When generated

This function is invoked by a TL to request an information packet transfer.

G.5.1.3 Effect of receipt

Upon receipt of a HIC_PACKET.request function, the source LL performs the following actions:

- Validates the parameters and verifies that the requested operation is possible.
- Segments the Data_Block into transmission sub-units which is an implementation dependent. LL determines these boundaries.
- Prefixes an appropriate Destination for the packet.
- Transfers the packet to the destination.
- Uses HIC_PACKET.confirmation to notify GPP that the information unit transfer has been successfully completed or abnormally terminated.

G.5.1.4 Function parameter usage

The Destination is a symbolic destination which shall be converted by the LL to an acceptable HIC destination terminal node.

The End_of_Packet_Marker_Type is managed by the LL. The control character used shall be EOP1.

The Data_Block parameter specifies the Information Unit to be transferred.

G.5.1.5 LL managed parameters

The following fields in the LL packet are managed by LL.

- Destination

- End_Of_Packet_Marker_Type

G.5.2 HIC_PACKET.indication

This function indicates the receipt of a single information unit at the destination terminal node for the TL.

G.5.2.1 Semantics

The figure below shows the semantics of this indication.

```
HIC_PACKET.indication (

        Residual_Destination,
        End_of_Marker_Type,
        Data_Block )
```

Figure 36 - HIC_PACKET.indication semantics

G.5.2.2 When generated

This function is generated after a packet is successfully received. The order that HIC_PACKET.indications will occur in the destination TL is the same order as the completed packets arrive successfully at the destination TL, which is not necessarily the order of transmission.

Upon receipt of portions of a packet the destination terminal node performs the following actions:

- Assembles the received tramsmission sub-units, recording any errors.

- When the packet is successfully received, HIC LL uses the HIC_PACKET.indication function to pass a completed packet to the GPP TL.

G.5.2.3 Effect of receipt

The receiving logical element may process a request to retrieve the payload of the packet.

G.5.2.4 LL managed parameters

The LL reports the values received as indicated as well as the transmission status.

The Transmission_Status parameter shall indicate status as one of the following:

- Successful - Packet sent to Recipient
- Unsuccessful - Packet was not sent completely due to abort or transfer error.

G.5.3 HIC_PACKET.confirmation

This function provides an appropriate response to a HIC_PACKET.request indicating the success or failure of the request.

G.5.3.1 Semantics

The figure below shows the semantics of this confirmation.

```
HIC_PACKET.confirmation (
        Transmission_Status )
```

Figure 37 - HIC_PACKET.confirmation semantics

G.5.3.2 When generated

This function is generated upon the completion of an attempt to transmit the packet.

G.5.3.3 Effect of receipt

The TL may prepare another request for the LL for the same port.

G.5.3.4 LL managed parameters

The LL shall assign the Transmission_Status value.

G.6 HIC events

HIC has one asynchronous event:

- the unexpected disconnect event.

This event establishes a condition which causes a logical element to perform certain actions. GPP devices normally expect HIC defined primitives to begin on its input after after the successful receipt of each Information Packet.

All other occurrences of an unexpected state cause the GPP device to report an unexpected disconnect event to the sending logical element. The receiving GPP device shall handle this as an unsuccessful Information Packet transfer condition.

G.6.2 LL unexpected disconnect event

If a complete information unit is not received according to HIC requirements, the receiving GPP device shall discard any portion of the information unit received and indicate an unsuccessful disconnect event. The receiving GPP device shall handle this as an unsuccessful Information Packet transfer condition.

G.6.3 LL unsuccessful Information Packet transfer condition

While processing an unsuccessful Information Packet transfer condition, the sending GPP device:

- may attempt to retransmit the Information Packet in error up to a limit determined by the sending LL.

- if operating as the Target for the task, the sending GPP device shall terminate the task after the specified number of retries by clearing all pending interface logical elements for the task and preparing sense data.

- if operating as the Initiator for a task, the sending GPP device shall abort the task. The Target prepares sense data. It is recommended that the initiator then request sense data from the Target if an autosense ILE is not provided by the Target for the task.

Annex H
(informative)

Heterogeneous InterConnect information transfer for GPP

An active serial interface consists of a continuous stream of encoded words transmitted along one fibre between two ends of a link. This stream of transmission words is

- between two in a point-to-point topology.

- between a Port and its network in a switched point-to-point topology.

- between two switches in a switched point-to-point topology.

The receiving port is not aware that it is about to receive an Information Packet. The sending port may not be aware of the present state of the destination GPP port.

The GPP information transfer protocol assumes delivery of complete Information Packets and recovers on that boundary when that assumption proves incorrect. That is, error free Information Packet delivery is not guaranteed at the time of transmission. GPP considers the network a transparent element in its service delivery subsystem protocol.

For each Information Packet received with a service delivery subsystem error for a packet, such as a parity or checksum error, the receiving GPP HIC port does not present any portion of the Information Packet to a GPP logical element (TL). The receiving Port does not process this information unit further and does not retain it.

H.1 Introduction to the GPP HIC packet protocol

Two GPP devices which require interaction must have a means to communicate:

- The lowest level HIC protocol involves individual packet transmission and reporting of buffer-to-buffer flow control (HIC CL and SL). This level of operation is transparent to GPP.

- The next level of HIC protocol is packet construction, transmission, and reassembly(HIC PL). This level of operation is transparent to GPP.

- The next level of HIC protocol is transaction management (TL). This level is equivalent to GPP services.

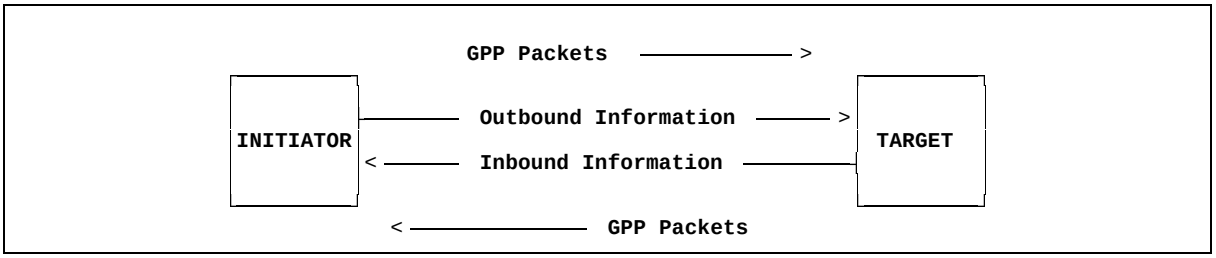


Figure 38 - Transaction Protocol for GPP over HIC

In GPP, an Information Packet is equivalent to one HIC packet. GPP provides the set of bytes in the form of an Information Packet to be transmitted by the LL, and the receiving LL provides a reassembled Information Packet

to the GPP logical element at the receiver. The content of the Information Packet is not known by, and therefore is not interpreted by, the LL. The actual packet transmission details are managed by LL. The GPP services is the lowest level of interaction between a GPP logical element and HIC.

H.2 GPP transaction protocol

This section specifies how GPP uses HIC and does not reflect the full capabilities of HIC.

H.2.1 HIC packets

A GPP Information Packet maps one-to-one with an HIC packet.

Packets may arrive at a destination in an order different than the order transmitted. Additionally, if a packet is received in error, it is discarded. GPP provides an internal sequencing mechanism within Information Packets to maintain the logical order of processing Information Packets within a task. This sequence control is not part of HIC since GPP is designed to use other interfaces, like HIC which may have no mechanisms for managing or detecting order of transmission. Also, GPP permits multiple paths to be used to transmit Information Packets for the same task.

H.2.2 Transport layer independence

All packets are fully self-identifying, relative to the Task with which they are associated. This lack of dependence on the physical transport system permits the multiple port option of GPP to be used with any type of Task spanning multiple Ports per terminal node within a single network or spanning two or more networks or other interfaces.

Annex I
(normative)

Asynchronous Transfer Mode (ATM)
AAL 5 usage for GPP

Asynchronous Transfer Mode (ATM) is an unconfirmed delivery service using cells, cell switching, and connections. ATM is closely allied with telephone systems, and much of the terminology is taken from there. Like IP, ATM depends on higher levels to form a reliable transmission service.

This annex and annex J specify the use of ATM as a transport layer for GPP. A logical system may be constructed using any of the service delivery subsystems in GPP or a combination of two or more.

ATM is normally used in conjunction with an isochronous physical layer, like Synchronous Optical Network (SONET). ATM does not require such a physical system when transferring data, although ATM/SONET seems, at times to be one acronym. Therefore, SONET is not a requirement for GPP when using ATM. Any available physical layer compatible with ATM is acceptable for use with GPP.

The basic concept of ATM is that small packets of information, called cells (53 bytes long), can be more easily managed and routed in the switches between nodes than can long, variable length frames. Cells can be switched with little or no buffering in the cell switches. The overhead is high with 5-9 bytes in each cell dedicated to the cell header and other overhead. Any additional protocol overhead is carried in the payload of a set of cells.

ATM has several options for sending data, but the one that most fits the requirements of SCSI is the Adaptation Layer 5 (AAL 5) which is meant for connection-oriented and connectionless data communication. The key feature of AAL 5 is that is does not need an isochronous physical layer to operate correctly. AAL 5 is a combination of obsolete layers 3 and 4.

Audio and video data usually require isochronous services and are managed with Adaptation Layers 1 and 2. These operations, while valuable, are not part of the SCSI model which does not specify any isochronous behavior.

Below SAR functions of the Adaptation Layer is the ATM layer. The ATM layer handles the final cell production steps. Below the ATM layer is the specific physical layer implemented for the node.

ATM has two interface levels: User-Network Interface (UNI) and Network-Network Interface (NNI). GPP services interacts only at the UNI level. All items in ATM related to the NNI are outside the scope of GPP. GPP interacts with the UNI level only [3]. The UNI specifies protocols to communicate with a telephone or campus switch. A GPP device shall operate as an ATM user connected to a private ATM switch. An ATM switch may or may not be in the system. ATM switches do not modify or otherwise transform the payload portion of the ATM cells.

I.1 ATM specific messages

There are no ATM specific messages defined for GPP.

I.2 ATM encapsulation of an information packet

ATM, and AAL 5 in particular, provides a mechanism to segment and reassemble byte streams that are longer then the cell size, called SAR. AAL 5 accepts packets up to 64 kilobytes in length, which is the maximum length permitted in GPP. Within ATM, an GPP Information Packet is called a Common Part Convergence Sublayer Protocol Data Unit (CPCS-PDU). The CPCS-PDU contains one Information Packet, AAL5 Pad of 0 to 44 bytes, and finally an 8-byte AAL 5 trailer. The total length of a CPCS-PDU shall be a multiple of 48 bytes.

The 8-byte AAL 5 trailer contains:

- CPCS-UU and CPI control fields. These control fields are not used by GPP.

- Packet Length contains the actual length in bytes of the Information Packet. The Packet Length value does not include the Pad or Trailer.

- CRC-32 provides a Cyclic Redundancy Check (CRC) over the entire CPCS-PDU, excluding the CRC field itself.

The CPCS-PDU is split into 48-byte SAR PDUs. Each SAR PDU is passed to the ATM layer that prefixes a 5 byte ATM header to complete the 53-byte cell. The overhead per cell is 5 bytes for each 48 bytes or 10.42 percent plus any overhead for AAL 5 Pad and the AAL 5 trailer.

A GPP Information Packet of 2048 bytes would be transmitted as 43 cells (2048+8+8/48) with the last cell padded with 8 fill bytes before the AAL5 trailer.

The Payload Type field (PTI) identifies the kind of payload as user or management data. GPP data always is identified as user data. GPP does not participate in the management data associated with ATM.

The other control function that affects GPP is the Cell Loss Priority (CLP) field in an ATM header. GPP cells should be set to request that cells not be discarded through the switches in the presence of congestion. This eliminates much of the error recovery. However, CLP is informational; if an ATM switch must discard a cell after having discarded all cell that indicate discard is acceptable, then a cell of this type will be discarded. Lost cells lead to incomplete Information Packets. As before, GPP services does not provide a partial Information Packet to a logical element.

I.3 ATM error control

Most errors will be detected as bit errors or discarded cells by ATM. Any CPCS-PDUs received with a CRD=C error or a mismatch in the Packet Length value versus the cells received shall be discarded with no indication to the destination GPP services.

I.4 ATM services

ATM depends on the inherent reliability of the transmission system and on higher layers to catch and correct errors other than cell loss and internal cell errors. ATM provides services to establish Virtual Paths between two endpoints in an ATM system. Each Virtual Path has associated with it an identifier (VPI) and a Virtual Path Connection (VPC). A VPC, in turn, consists of one or more Virtual Channel Connections (VCCs) that describe point-to-point physical links in the Virtual Path.

A Virtual Channel Connection (VCC) consists of two unidirectional transport services, flowing in opposite directions between the two endpoints. Each unidirectional service is called a Virtual Circuit (VC). Each GPP Information Packet flows in only one VC.

GPP uses the logical connection of a VCC for its activity. ATM may collect two or more VCCs into a VPC. Creating, managing and deleting a VPC are outside the scope of GPP services. A separate VCC may be needed in the UNI to manage setting up and terminating VCCs used for GPP Information Packets. This is an implementation issue with the ATM implementation available to the system.

ATM AAL 5 provides both a connection-oriented service and a connectionless service. Within these two services, there are two modes:

- Message mode - In this mode a single Information Packet is transmitted as a set of cells. This mode shall be used by GPP.

- Streaming mode - In this mode, a continuous stream of fixed size blocks is transmitted. This is not appropriate for GPP operations and shall not be used.

I.5 ATM support of SAM

ATM has well defined link level responses and control processes to assure detection of a lost cell between an Initiator and a Target.

- ATM specifies the exact manner in which a cell shall be transmitted. GPP specifies no new responses or control requests other than those specified for ATM. No equivalent vendor unique response types or field values shall be specified.

- Logical responses to the content of an Information Packet are generated by the receiving logical element, where applicable, as new Information Packets.

The connection-oriented service of ATM AAL 5 provides strict ordering of the delivery of information within a VCC between two hosts permits ordered and head of queue operations as well as simple queue management operations to behave as specified in SAM. The strict ordering also causes Task Management functions to arrive in the correct order as sent by the Initiator of a task. That is, the state of a logical unit is approximately the same as would be observed on a SCSI Parallel bus if the same stream of information were transferred in the same order. This provides remote device behavior similar to that of a local device.

This strict ordering capability is not available if the connectionless service of AAL 5 within a VCC are used. If multiple VCCs are in use between endpoints the strict ordering control is also not available.

The SAR PDU CRC provides a complete CRC check across each Information Packet. If a cell is lost, the Information Packet is discarded at the receiving endpoint by GPP services.

I.6 Service interface from GPP to ATM

The service interface for an PDU is specified in the appropriate ATM documents. GPP is an upper layer protocol to ATM AAL 5. The service protocol currently specified for AAL 5 shall be used. Invoking these services causes an Information Packet encapsulated in by AAL 5 to be transmitted to a destination. Because of the potential distances and time involved for transmission, a cell streaming protocol is used in ATM. This protocol is used by GPP to provide a performance enhancement. This streaming protocol is consistent with the Information Packet prefetch operations specified in the body of GPP.

ATM does not specify a precise protocol or interface from the application to ATM. Each platform has its own internal architecture which must be observed. This might be compared to the interface for SCSI host adapter cards. Each vendor has the right to specify an interface. However, over time, there is some convergence on two or three major

interfaces. GPP services is the component of GPP responsible to match the ATM interface and not the application client or the logical unit.

Since ATM provides a connection-oriented service there is a series of steps to set up, use and tear down a connection. The general sequence of events is outlined below:

- Establish the connection - This uses a VCC between the user and the network to set up the VCC with the destination. When this protocol completes successfully, a connection is established and GPP Information Packets may be transmitted.

- Established connection - At this time GPP Information Packet transfers occurs. If either user needs to break the connection, that user starts a close process.

- Closing the connection - During this process, one of the users requests to close the connection. This begins a sequence of events that breaks the connection.

When the connection is closed, no further GPP Information Packets can be transferred until another VCC is established. It is not considered a GPP error for ATM or GPP services to close the connection. A new connection may be established to continue operations as required.

If an error terminates a connection other than specified above, each user is notified. If a partial Information Packet has been received at the application, it shall be discarded. All buffers in ATM at both users are released. This event only breaks the connection and does not cause either logical element to abort any tasks. That is, a ATM connection loss has no effect on logical operations between the GPP devices.

The exact method of invoking these services depends on the host platform and the ATM implementation available and is outside the scope of GPP. See the system documentation for precise details.

Annex J
(informative)

Asynchronous Transfer Mode (ATM) AAL 5 information transfer for GPP

The Asynchronous Transfer Mode (ATM) system is a relatively new transmission system based on the aggregation of circuit switched and frame relay work of the past. A basic premise of the ATM is that the transport system and switches are highly reliable. Based on telecommunications protocols, ATM is aimed at isochronous services. However, the long-haul aspects of ATM and the transfer rates in a Virtual Circuit of up to 600 Mb/s caught the eye of data communication users who need long-haul services. With GPP mapped to this system, any user can communicate with any other user based on a quality of service agreement. ATM is a set protocols that specifies how an application is to use ATM and its underlying physical layers to communicate with another application in a remote host.

Outside the host to network interface, ATM has additional protocols for mapping between switches. This level of protocols is beyond the scope of GPP.

J.1 GPP use of ATM

GPP concentrates on the interface between the GPP services (i.e., the application) and ATM AAL 5. This focus means that the logical unit and application client of SAM are not concerned with the details of the process of transmission. GPP retains the separation of these components from the service delivery subsystem. This limits the focus for definition and implementation to a much smaller region of the complete ATM protocols.

GPP focuses on the AAL 5 layer since it provides the level of data delivery reliability expected in SCSI. Because of the nature of the ATM, a logical connection is established between the two cooperating nodes. After the logical connection is established, each node may expect Information Packets from the other. If a logical connection is not established, the Information Packets are not sent; another connection attempt may be made. This process is specified in the appropriate ATM documents. The receiving port is not aware, other than that a logical connection exists, that an Information Packet is about to be received. The sending port may not be aware of the present state of the destination port.

GPP assumes the delivery of complete Information Packets and recovers on that boundary when an error occurs. ATM is unaware of the boundary between Information Packets since ATM manages a stream of cells. The closest knowledge ATM has is that it is operating in message mode. In message mode, one PDU maps to a contiguous set of one or more cells to be transmitted to the destination user.

This implies that the source and destination applications are synchronized on the VCs (i.e., one in each direction) between them for each VCC. The structure of GPP Information Packets provides the destination application with a length indication for each Information Packet. The destination application uses this field to mark the ends of Information Packets. ATM, using its message, also marks the beginning and end of each PDU. Since a PDU is equal to an Information Packet, one message is equal to one Information Packet.

J.2 Error Management

Error free Information Packet delivery is not guaranteed at the time of transmission. The ATM processes do not provide for positive responses from the opposite end of a logical connection. GPP is responsible to generate appropriate responses according to the available GPP services resources. A partially received Information Packet is not be processed by the GPP application and is discarded by the receiving GPP application. GPP services is the component responsible to provide only complete Information Packets to the receiving application.

If an Information Packet is mapped into exactly one cell, that cell might be discarded due to congestion. Within a Task, loss of an Information Packet is detected by an omission in the Sender's Sequence Number in the Information Packets in either direction.

Single SCSI commands like a 6-byte READ command require only 32 bytes in an Information Packet. This command might never arrive at the Logical Unit, but this is detected by a timeout on the entire Task by the application client

There is no data dependency between the ATM or SAR headers and the GPP Information Packet content. The ATM and SAR headers are not to be processed by a receiving GPP application. The only level of interaction is isolated to GPP services. Only the net payload, an Information Packet, is to be processed by a GPP application.

The VCC between two GPP devices is a full-duplex, fully operational bi-directional transmission path. The actual mechanisms used by ATM to manage this method of operation is beyond the scope of GPP. The ATM VCC between two users may be used for more than one Task. A separate VCC need not be established for each Task. GPP isolates the logical content from the service delivery subsystem.

This independence from the service delivery subsystem permits GPP to use cooperating sets of ports, two or more pairs to accomplish a single Task. Multiple port pairs, when available, may be used to increase the usable bandwidth between the nodes as well as to provide a highly available system environment.

GPP services is responsible to start and stop connections and manage the flow of complete Information Packets between application client and logical unit. If a connection is terminated for any reason, the receiving GPP services discards any partial Information Packet(s). The source GPP services is also notified of any termination of a connection. If any Information Packets have not been fully acknowledged, they are made available for retransmission when a connection is reestablished. Of course, if a timeout occurs between the source logical element and the source GPP services, this is reported to the logical element in the normal manner specified in clause 8.

Index

Page

End of Document

Printed:

May 31, 1995 at 08:56pm