INTERNATIONAL STANDARD

ISO/IEC 9314-6

First edition
1998-08

Information technology –
Fibre distributed data interface (FDDI) –

Part 6:
Station Management (SMT)

Reference number
ISO/IEC 9314-6:1998(E)

INTERNATIONAL STANDARD

ISO/IEC 9314-6

First edition
1998-08

Information technology – Fibre distributed data interface (FDDI) –

Part 6: Station Management (SMT)

PRICE CODE XD

For price, see current catalogue

CONTENTS

Tables

Figures

FOREWORD

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

International Standard ISO/IEC 9314-6 was prepared by Joint Technical Committee ISO/IEC JTC 1 *Information technology*, Subcommittee SC 25*, Interconnection of information technology equipment.*

ISO/IEC 9314 consists of the following parts, under the general title *Information technology – Fibre Distributed Data Interface (FDDI):*

– *Part 1: Token Ring Physical Layer Protocol (PHY) (1989)*
– *Part 2: Token Ring Media Access Control (MAC) (1989)*
– *Part 3: Physical Layer Medium Dependent (PMD) (1990)*
– *Part 4: Single Mode Fibre Physical Layer Medium Dependent (SMF-PMD)* [1)]
– *Part 5: Hybrid Ring Control (HRC) (1995)*
– *Part 6: Station Management (SMT)*
– *Part 7: Physical Layer Protocol (PHY-2)*
– *Part 8: Media Access Control-2 (MAC-2)*
– *Part 9: Low-Cost Fibre – Physical Medium Dependent (LCF-PMD) (under consideration)*
– *Part 10: Token Ring Twisted Pair Physical layer Medium Dependent (TP-PMD) (under consideration)*
– *Part 13: Conformance Test Protocol Implementation Conformance Statement Proforma (CT-PICS)*
– *Part 20: Physical Medium Dependent Conformance Testing (PMD-ATS) (under consideration)*
– *Part 21: Physical Layer Protocol Conformance Testing (PHY-ATS) (under consideration)*
– *Part 25: Abstract Test Suite for FDDI – Station Management Conformance Testing (SMT-ATS)*
– *Part 26: Media Access Control Conformance Testing (MAC-ATS) (under consideration)*

1) To be published.

INTRODUCTION

The Fibre Distributed Data Interface (FDDI), ISO/IEC 9314, is intended for use in a high-performance general purpose multi-node network and is designed for efficient operation with a peak data rate of 100 Mbit/s. It uses a Token Ring architecture with optical fibre as the transmission medium. FDDI provides for hundreds of nodes operating over an extent of tens of kilometres.

Station Management (SMT) specifies the local portion of the system management application process for FDDI, including the control required for proper operation of a node in an FDDI ring. SMT provides services such as connection management, station insertion and removal, station initialization, configuration management, fault isolation and recovery, communications protocol for external authority, scheduling policies, and collection of statistics.

When the set of basic FDDI standards, ISO/IEC 9314, is completed it will include the following standards:

a) A Media Access Control (MAC), which specifies the lower sublayer of the Data Link Layer of ISO/IEC 9314,

b) A Physical Layer Media Dependent (PMD), which specifies the lower sublayer of the Physical Layer of ISO/IEC 9314,

c) A Physical Layer Protocol (PHY), which specifies the upper sublayer of the Physical Layer of ISO/IEC 9314.

A number of extensions to ISO/IEC 9314 are completed or in process. One extension, ISO/IEC 9314-5, for Hybrid Ring Control (HRC), commonly known as FDDI-II, extends the capability of FDDI to handle isochronous data streams at a multiplicity of data rates. Another extension, ISO/IEC 9314-4, provides for a single-mode optical fibre version of PMD (SMF-PMD) and will permit optical links of up 60 km.

Other extensions, addressing alternate PMDs, will provide low-cost attachments for use in concentrator-to-workstation environments.

This part of ISO/IEC 9314 for SMT represents the final standard in the set of basic FDDI standards. SMT is a sophisticated document specifying many critical aspects of interoperability in a multi-vendor FDDI network and, as such, has proved to be by far the most difficult of the set of FDDI standards to complete. The successful completion of the work on SMT is the result of a high degree of cooperation between competing manufacturers of FDDI equipment.

INFORMATION TECHNOLOGY — FIBRE DISTRIBUTED DATA INTERFACE (FDDI) —

Part 6: Station Management (SMT)

1 Scope

This part of ISO/IEC 9314 specifies the Station Management (SMT) for the Fibre Distributed Data Interface (FDDI).

FDDI provides a high bandwidth (100 megabits per second) general purpose interconnection among computers and peripheral equipment using optical fibre as the transmission medium in a ring configuration. FDDI can be configured to support a sustained transfer rate of approximately 80 megabits (10 megabytes) per second. The use of dual attachment stations with dual MACs allows these rates to be doubled under the circumstance of a fault-free FDDI ring.

FDDI establishes the connection among many stations (nodes) distributed over distances of several kilometres in extent. Default values for FDDI were calculated on the basis of 1 000 physical connections and a total fibre path length of 200 km.

The FDDI consists of

a) A Physical Layer (PL), which provides the medium, connectors, optical bypassing, and driver/receiver requirements. PL also defines encode/decode and clock requirements as required for framing the data for transmission on the medium or to the higher layers of the FDDI. For the purposes of this part of ISO/IEC 9314, references to the PL are made in terms of the Physical Layer protocol (PHY) and the Physical Layer Media Dependent (PMD) entities which are the upper and lower sublayers of PL, respectively.

b) A Data Link Layer (DLL) which controls the accessing of the medium and the generation and verification of frame check sequences to assure the proper delivery of valid data to the higher layers. DLL also concerns itself with the generation and recognition of device addresses and the peer-to-peer associations within the FDDI network. For the purposes of this part of ISO/IEC 9314, references to the DLL are made in terms of the Media Access Control (MAC) entity which is the lowest sublayer of DLL.

c) A Station Management (SMT) standard, this part of ISO/IEC 9314, which provides the control necessary at the station (node) level to manage the processes underway in the various FDDI layers such that a station may work cooperatively as a part of an FDDI network. SMT shall provide services such as connection management, station insertion and removal, station initialization, configuration management, fault isolation and recovery, communications protocol for external authority, scheduling policies, and collection of statistics.

The definition of SMT as contained herein includes the set of services that it provides for, and receives from, the other entities that are contained within a node. Within SMT resides both knowledge of the uniqueness of this node and the current network structure to the extent that this node's function is affected.

The set of International Standards for FDDI, ISO/IEC 9314, specifies the interfaces, functions and operations necessary to insure interoperability between conforming FDDI implementations. This part of ISO/IEC 9314 is a functional description. Conforming implementations may employ any design technique which does not violate interoperability.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ISO/IEC 9314. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO/IEC 9314 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

ISO/IEC 7498-4(1989) *Information processing systems – Open Systems Interconnection – Basic Reference Model – Part 4: Management framework*

ISO 9314-1: 1989, *Information processing systems – Fibre Distributed Data Interface (FDDI) – Part 1: Token Ring Physical Layer Protocol (PHY)*

ISO 9314-2: 1989, *Information processing systems – Fibre Distributed Data Interface (FDDI) – Part 2: Token Ring Media Access Control (MAC)*

ISO/IEC 9314-3(1990) *Information processing systems – Fibre Distributed Data Interface (FDDI) – Part 3: Physical Layer Medium Dependent (PMD)*

ISO/IEC TR3 8802-1(1997) *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 1: Overview of Local Area Network Standards*

ISO/IEC 8824: 1990, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*

ISO 8825: 1990, *Information technology – Open Systems Interconnection – Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)*

ISO/IEC 10165-4: 1992, *Information technology – Open Systems Interconnection – Structure of management information – Part 4: Guidelines for the definition of managed objects*

3 Definitions

For the purposes of this part of ISO/IEC 9314, the following definitions apply. In some cases these definitions may duplicate those contained in other parts of ISO/IEC 9314. Such definitions are included for completeness and to improve readability. In certain cases, definitions herein may slightly update those contained in the earlier published parts of ISO/IEC 9314 to improve their clarity. Other parts of ISO/IEC 9314, e.g. FDDI MAC, PHY and PMD, may however, contain additional definitions of interest.

3.1 attachment: The capability of a station or concentrator for connection into an FDDI network. Stations and concentrators are classified as dual attachment, single attachment or null attachment.

3.2 bypass: The ability of a node to optically isolate itself from the FDDI network while maintaining the continuity of the cable plant.

3.3 concentrator: An FDDI node that has additional Ports beyond those required for its own attachment to an FDDI network. These additional Ports (type M – see 5.2.4) are for attaching other FDDI nodes (including other concentrators) in a tree topology.

3.4 counter-rotating: An arrangement whereby two signal paths, one in each direction, exist in a ring topology.

3.5 Dual Attachment Concentrator (DAC): A concentrator that offers a dual attachment to the FDDI network and is capable of accommodating a dual (counter-rotating) ring.

3.6 Dual Attachment Station (DAS): A station that offers a dual attachment to the FDDI network and is capable of accommodating a dual (counter-rotating) ring.

3.7 dual ring (FDDI dual ring): A pair of counter-rotating logical rings.

3.8 entity: An active service or management element within an Open Systems Interconnection (OSI) layer, or sublayer.

3.9 fibre optic cable: A cable containing one or more optical fibres.

3.10 Local Path: A Local Path represents the segment(s) of ring(s) other than the primary ring and secondary ring that pass through the node.

3.11 logical ring: The set of MACs serially connected to form a single ring. A fault-free FDDI network provides two logical rings.

3.12 Media Interface Connector (MIC): A mated connector pair that provides an attachment between an FDDI node and a fibre optic cable plant. The MIC consists of two parts: a MIC plug and a MIC receptacle.

3.13 MIC plug: The male part of the MIC which terminates a fibre optical cable.

3.14 MIC receptacle: The female part of the MIC which is contained in an FDDI node.

3.15 network (FDDI network): A collection of FDDI nodes interconnected to form a trunk, or a tree, or a trunk with multiple trees. This topology is sometimes called a dual ring of trees.

3.16 node: A generic term applying to an active element in an FDDI network (station or concentrator).

3.17 Null Attachment Concentrator (NAC): A concentrator that does not contain an A, B, or S Port.

3.18 Path: A Path represents the segment(s) of a logical ring that pass through a node.

3.19 Physical Connection: The full-duplex physical layer association between adjacent PHY entities (in adjacent nodes) in an FDDI network, i.e. a pair of Physical Links.

3.20 Physical Link: The simplex path (via PMD and attached medium) from the transmit function of one PHY entity to the receive function of an adjacent PHY entity (in adjacent nodes) in an FDDI network.

3.21 Port: A PHY entity and a PMD entity in a node, together creating a PHY/PMD pair, that may connect to the fibre media and provide one end of a physical connection with another node.

3.22 Primary Path: A Primary Path represents, to the best of a node's knowledge, the segment(s) of the primary ring that pass through the node. Conditions may exist in parts of the network which may cause the Path to be in a different ring (e.g. Secondary Path instead of Primary Path).

3.23 primitive: An element of the services provided by one entity to another.

3.24 receiver (optical): An opto-electronic circuit that converts an optical signal to an electrical logic signal.

3.25 repeater: A physical-layer relay in an FDDI network. A repeater is not further defined in this International Standard.

3.26 ring: A set of nodes wherein information is passed sequentially between nodes, each node in turn examining or copying the information, finally returning it to the originating node.

3.27 rooted node: A node that does not have any active A, B, or S Ports in tree mode.

3.28 Secondary Path: A Secondary Path represents, to the best of a node's knowledge, the segment(s) of the secondary ring that pass through the node. Conditions may exist in parts of the network which may cause the Path to be in a different ring (e.g. Primary Path instead of Secondary Path).

3.29 services: The services provided by one entity to another. Data services are provided to a higher layer entity; management services are provided to a management entity in the same or another layer.

3.30 Single Attachment Concentrator (SAC): A concentrator that offers a single attachment to the FDDI network.

3.31 Single Attachment Station (SAS): A station that offers a single attachment to the FDDI network.

3.32 station: An addressable node on an FDDI network capable of transmitting, repeating and receiving information. A station has exactly one SMT, at least one MAC, at least one PHY, and at least one PMD.

3.33 symbol: The smallest signalling element used by the Data Link Layer (DLL). The symbol set consists of 16 data symbols and eight control symbols.

3.34 transmitter (optical): An opto-electronic circuit that converts an electrical logic signal to an optical signal.

3.35 trunk: A physical loop topology, either open or closed, employing two optical fibre signal paths, one in each direction (i.e. counter-rotating), forming a sequence of peer connections between FDDI nodes. When the trunk forms a closed loop it is sometimes called a trunk ring.

3.36 tree: A physical topology consisting of a hierarchy of master-slave connections between a concentrator and other FDDI nodes (including subordinate concentrators).

4 Conventions and abbreviations

4.1 Conventions

The terms SMT, MAC, LLC, PHY, and PMD when used without modifiers, refer specifically to the local entities.

The terms node, station, concentrator and repeater are used as follows in this part of ISO/IEC 9314. The term node is used as a generic term to denote any active element in an FDDI network. Station is used to denote a node that has at least one MAC. Concentrator is used to denote any node that has concentrator capability. The terms station and concentrator thus overlap in such a way that some nodes may be referred to by either term. In this case, the term actually used will be dependent upon the context of the usage. The term repeater is used only to denote a physical-layer relay in an FDDI network and is not further defined.

This Intentional Standard when referring to any of the PMD components assumes the multimode optical fibre PMD. This does not preclude the use of other PMD media types.

Low lines (e.g. requested_service_class) are used as a convenience to form the name of signals, functions, etc., which might otherwise be misinterpreted as independent individual words if they were to appear in text.

In 6.4, hyphens are used in place of low lines. This is done to maintain compatibility of the encoding of the names of attributes with the standard on ASN.1 (see ISO 8824) which does not allow the use of low lines. No difference in meaning is implied. Thus T-Max in 6.4 is exactly equivalent to T_Max as used in clause 10 of this International Standard and in ISO/IEC 9314-2.

The use of a period (e.g. SM_MA_UNITDATA.request) is equivalent to the use of a low line except that a period is used as an aid to distinguish modifier words appended to an antecedent expression.

The use of an asterisk (e.g. *:SM_PM_CONTROL.request) indicates that the primitive is to be sent to all of the entities of the type that receive the primitive. Thus, issuing *:SM_PM_CONTROL.request will send a SM_PM_CONTROL.request to all the PMD entities under control of this SMT.

In the presentation of diagrams, dashed lines are used to indicate optional entities, data paths, transitions and states. Dotted lines are used to indicate a functional unit that may be broken into other functional units.

Timers are given by a name of the form TXX, where XX are two capital letters. An example is the PCM timer, TPC.

This part of ISO/IEC 9314 on FDDI SMT contains four kinds of documentation as follows:

a) Narrative text, including text associated with state machines,

b) State machine diagrams, including associated footnotes,

c) Pseudo code,

d) Examples, which are specifically noted as such.

If any discrepancies exist between the above, the following precedence shall be used to resolve those discrepancies and determine conformance:

a) State machine diagrams,

b) Pseudo code,

c) Narrative text.

Examples are provided only for clarification and shall not be used for determining conformance.

4.1.1 State machines

SMT operation is defined using cooperating state machines. It is assumed that time elapses only in discrete states, with instantaneous transitions between the states.

State diagrams are expressed using vertical staffs to represent states and horizontal arrows to represent transitions.

Transitions are illustrated with the triggering condition located above the horizontal arrow and any actions on transition located below the transition arrow. Transition actions are performed while remaining in the previous state, before entry into the new state.

The state name appears above the vertical staff representing state and entry actions in the state, if any, appear directly below the state name. Entry actions in a state are executed on any transition entering a state, including those transitions exiting and entering that same state.

The following event-processing sequence is assumed:

a) Evaluate all transition conditions from the current state;

b) If a transition condition is satisfied, then

   1) perform the associated transition actions in the current state,

   2) enter the new state,

   3) perform entry actions, if any, for the new state,

   4) if a transition condition from this new state is satisfied, then repeat the transition sequence steps 1 to 4;

c) Perform any instate actions when their associated conditions are satisfied.

Logical symbols are represented in state machines and pseudo code using | to represent the 'or' operator and & to represent the 'and' operator.

Footnotes are used in state diagrams to give precise detail on transition conditions, transitions actions, entry actions, and instate actions

4.1.2 Default and initial values

Default values are defined for many of the attributes and parameters in this part of ISO/IEC 9314. These default values provide correct protocol operation and interoperability of equipment over a broad range of network configurations. Implementers may use alternate values than the defaults specified. The effects of using an alternate value are not directly described.

An initial value is the value assumed by the attribute or parameter as the result of an implementer defined causative action (e.g. power-on). When an initial value is specified in this part of ISO/IEC 9314, the station shall be capable of assuming that value.

4.2 Abbreviations

A	Port type A
ALS	Active Line State
ASN.1	Abstract Syntax Notation One
B	Port type B
BER	Basic Encoding Rules
CCE	Configuration Control Element
CEM	Configuration Element Manager
CF_State	Configuration state
CFM	Configuration Management
CMT	Connection Management
DA_Flag	Duplicate Address flag
DAC	Dual Attachment Concentrator
DAS	Dual Attachment Station
DM-DAS	Dual-MAC Dual Attachment Station
DNA	Downstream Neighbour Address
ECF	ECHO Frame
ECM	Entity Coordination Management
ESF	Extended Service Frame
GDMO	Guidelines for the Definition of Managed Objects
HLS	Halt Line State
ID	Identifier (field)
ILS	Idle Line State
LCT	Link Confidence Test
LEM	Link Error Monitor
LER	Link Error Rate
LS_Flag	Line State flag
LSU	Line State Unknown
M	Port type M
MIB	Management Information Base
MLS	Master Line State
MSB	Most Significant Bit
N	PC_Mode is None
NAC	Null Attachment Concentrator
NIF	Neighbour Information Frame
NLS	Noise Line State
NSA	Next Station Addressing
P	Mode is Peer
Path_Test	Path Test state variable
PC_Withhold	PCM Connection Withhold variable
PCM	Physical Connection Management
PDR	PHY Data Request
PDU	Protocol Data Unit
PMD	Physical Media Dependent
PMF	Parameter Management Frame
QLS	Quiet Line State

R_Val	PCM signalling Received Value array
RAF	Resource Allocation Frame
RDF	Request Denied Frame
RMT	Ring Management
S	Port type S
SAC	Single Attachment Concentrator
SAS	Single Attachment Station
SBA	Synchronous Bandwidth Allocation
SIF	Status Information Frame
SM-DAS	Single-MAC Dual Attachment Station
SRF	Status Report Frame
T	Mode is Tree
T_Val	PCM signalling Transmitted Value array
TEC	Timer, Entity Coordination
TID	Timer, Idle Detection
TNE	Timer, Noise Events
TNN	Timer, Neighbour Notification
TPC	Timer, Physical Connection
TRM	Timer, Ring Management
TSR	Timer, Status Reporting
TVD	Timer, Valid Downstream Neighbour
TVU	Timer, Valid Upstream Neighbour
UNA	Upstream Neighbour Address
UNDA_Flag	Upstream Neighbour Duplicate Address flag
WC_Flag	Withhold Connection flag

5 General description

An FDDI network consists of a set of nodes logically connected. Information is transmitted sequentially, as a stream of suitably encoded symbols, from one active node to the next. Each node generally regenerates and repeats each symbol and serves as the means for attaching one or more devices to the network for the purpose of communicating with other devices on the network. The method of actual physical attachment to an FDDI network may vary and is dependent on specific application requirements as described herein.

The basic building block of an FDDI network is the Physical Connection which consists of paired Physical Layer entities in two adjacent nodes on the FDDI network connected with a transmission medium. Connection to the physical medium, specified in PMD, is controlled by the station insertion and removal algorithms of Station Management (SMT) which are contained herein.

SMT specifies the local functions within an FDDI node necessary to manage the FDDI network. This clause describes the physical and logical topologies of an FDDI network and specifies examples of allowable node configurations used for specification of SMT protocols.

5.1 Definition of an FDDI node

A variety of internal node configurations are possible. However, a node shall have one, and only one, SMT entity. It may, however, have multiple instances of MACs, PHYs, and PMDs, with the actual number, within bounds, being implementer defined.

Several internal node configurations are defined within SMT with state machines provided to specify their operation. These fall into the general classifications of single attachment nodes and dual attachment nodes. FDDI trunk rings are normally composed of dual attachment nodes which have two Ports (each consisting of one PHY and one PMD) to accommodate the dual (counter-rotating) rings.

Concentrators, which may be single attachment, dual attachment, or form the root of a tree of other FDDI nodes (null attachment), are also defined. Concentrators provide for the connection of single attachment nodes (either stations or concentrators) which have only one Port and therefore cannot directly attach to the dual ring. A dual attachment station or concentrator may also be connected to a concentrator for use as the functional equivalent of the corresponding single attachment node (or nodes if Port A and Port B are both connected to concentrators).

A single attachment node requires a minimum of one Port. A dual attachment node requires a minimum of two Ports. To be designated a station, a node requires a minimum of one MAC. Stations may have additional MACs, with MACs inserted in each of the dual rings in the case of dual attachment stations. A concentrator requires one additional Port [type M (see 5.2.4)] for each tree connection that is to be provided. Different concentrator types may or may not have any MACs. The absence of a MAC may limit the functionality of the concentrator. If it has at least one MAC, a MAC shall be located at the exit Port(s) (i.e. a MAC follows the stations inserted on its M Ports). In general, no maximum is placed on the number of MACs that may be contained within a station or concentrator.

The six specific station and concentrator types listed below are used in describing SMT configuration and topology protocols. Figures 1 to 3 depict examples of some of these. Other internal node configurations that are interoperable with these six node types are not precluded by this part of ISO/IEC 9314.

Single MAC – Dual Attachment Station (SM-DAS)
Dual MAC – Dual Attachment Station (DM-DAS)
Single Attachment Station (SAS)
Dual Attachment Concentrator (DAC)
Single Attachment Concentrator (SAC)
Null Attachment Concentrator (NAC)

FDDI nodes are differentiated by PMD entities. Each PHY/PMD pair, designated a Port, belongs to one of four types: A, B, M, or S. The type of a PMD may be indicated by the keying of the MIC receptacle (see the appropriate part of ISO/IEC 9314 on PMD).

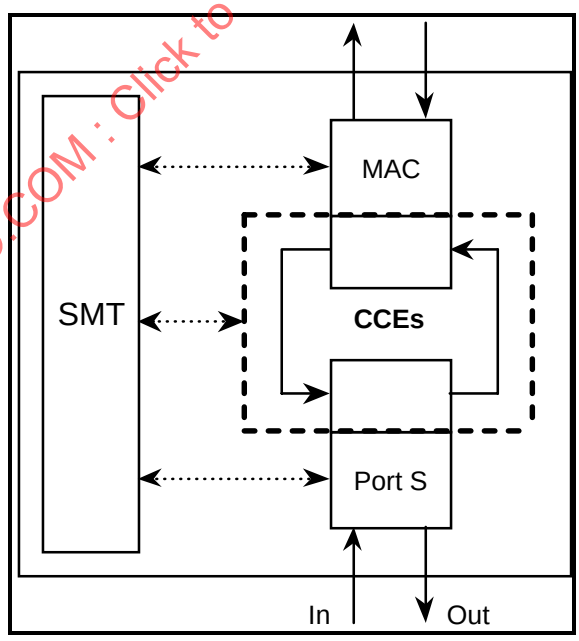


Figure 1 – Example Single Attachment Station (SAS)

Dual attachment nodes (DAS and DAC) may attach directly to the trunk ring. Each dual attachment node contains two Ports designated as A and B. Port A is intended to be connected to the primary ring on the incoming fibre and the secondary ring on the outgoing fibre. Similarly, Port B is intended to be connected to the incoming fibre of the secondary ring and the outgoing fibre of the primary ring. Therefore, a properly formed trunk ring is composed of a set of stations with the Port A of one station connected to the Port B of the neighbouring station.

Concentrator nodes (DAC, SAC, and NAC) contain one or more Ports of type M to provide connections within the concentrator tree.

A single attachment node (SAS or SAC) has a Port of type S which is intended to be attached to a Port of type M within a concentrator tree.

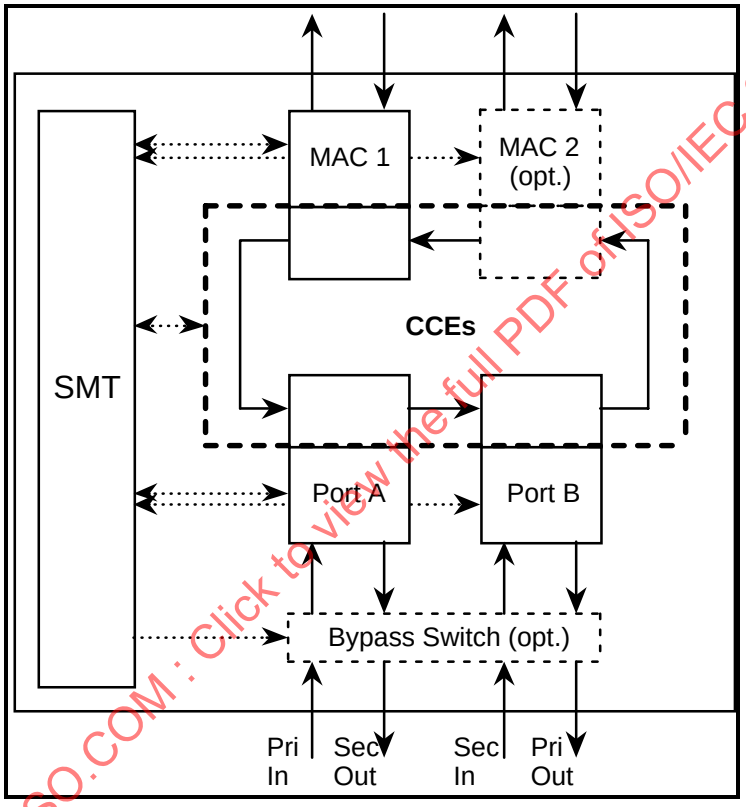


Figure 2 – Example Dual Attachment Station (DAS)

Given that a node may have a variety of internal configurations, with multiple MACs, PHYs, and PMDs, consistent external behaviour shall be required to claim conformance with ISO/IEC 9314. At the discretion of an implementer, a node may be considerably more complex in its internal structure than the example configurations.

This part of ISO/IEC 9314 does not specify implementation. Any node, regardless of its internal implementation, that conforms at its external interfaces to the requirements defined in this part of ISO/IEC 9314 is permitted.

5.2 Definition of an FDDI network

The FDDI network topology may be viewed at two distinct levels: physical and logical. Physical topology describes the arrangement and interconnection of nodes with physical connections. In contrast, logical topology describes the paths by which tokens and information flow through the network between MACs.

An FDDI network forms a dual ring of trees or a subset of a dual ring of trees physical topology. The implication on the logical topology is that at most two logical sets of MAC entities (i.e. two independent token/data Paths) exist in a single, fault-free FDDI network. A set of MAC entities that are in the same token/data Path is called a logical ring hereafter. The valid physical topologies are also constrained to those where physical connections and internal configurations in nodes form at most two logical rings.

The logical and physical topologies of an FDDI network are not necessarily the same. The tree structure provided through concentrators can have the token Path entering and exiting the concentrator many times on the same ring, where logically, the concentrator may only appear on the ring once. Also, the number of MACs and attachments in a station need not be equal; so, a station in the trunk ring is physically in both rings but may be in only one of the logical rings.

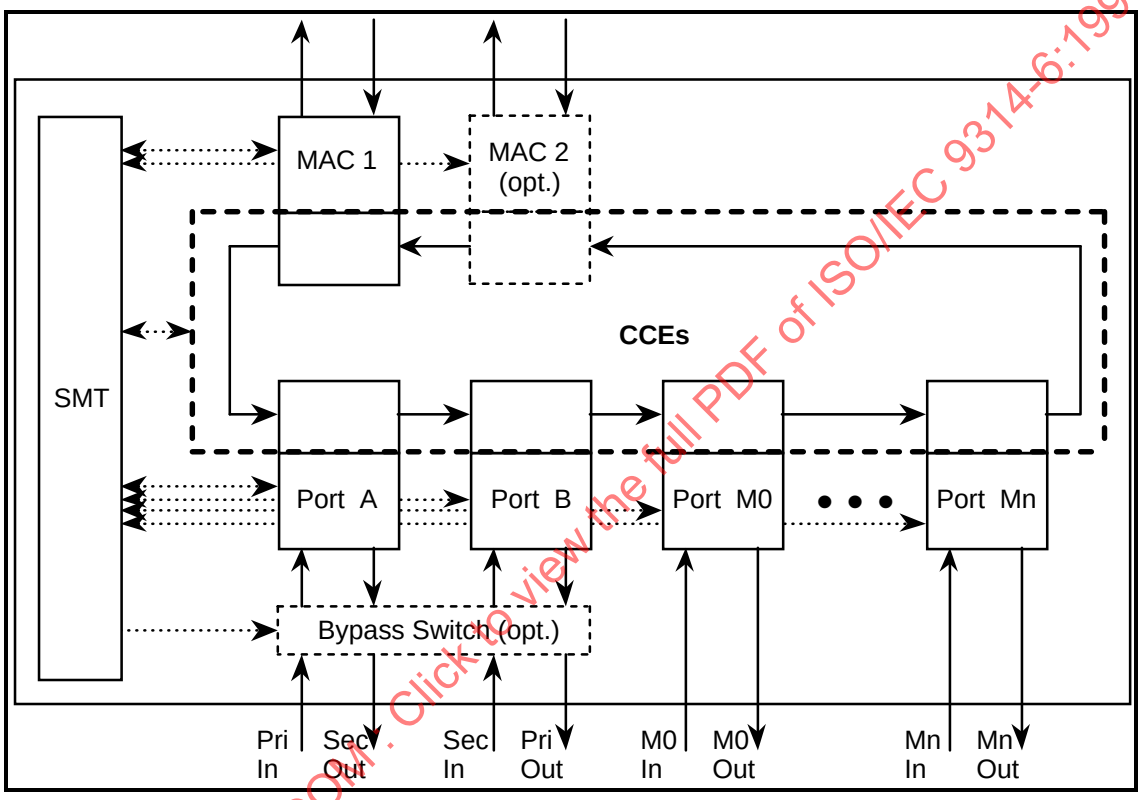


Figure 3 – Example Dual Attachment Concentrator (DAC)

5.2.1 Physical topology

All physical connections in an FDDI topology are duplex links. In a fully connected trunk ring, a duplex link supports counter-rotating rings; and in a tree, the duplex link provides transmit and receive paths for one of the dual rings. Combining these two structures in an FDDI topology, as shown in figure 4, produces a dual ring of trees in a fully connected network. The nodes in the network are interconnected so that at most one dual ring exists.

By removing physical connections and nodes from any legal FDDI topology, one or more legal FDDI topologies are formed. Subsets of legal topologies are legal topologies, so that failure of nodes or connections in an FDDI network produce legal topologies.

A subset of any legal topology is also a legal topology. An FDDI network can be configured in several different physical topologies:

A dual ring with trees
A dual ring without trees
A wrapped ring with trees
A wrapped ring without trees
A single tree

5.2.2 Logical topology

At the logical level, a single legal FDDI topology consists of at most two separate logical rings, designated as the primary and secondary rings. The two logical rings are formed from the pairs of opposing physical links that make up the Physical Layer connections. A set of Dual Attachment Stations connected into a closed loop form two counter-rotating rings, referred to as a dual ring. Each of these counter-rotating rings is the basis of a logical ring. If a closed loop of physical connections is not present (i.e. the trunk ring is wrapped) in an FDDI topology, then only one logical ring is present.

Recovery from a single fault on the trunk ring is achieved by joining the two rings in the two nodes adjacent to the fault to create a single logical ring. This wrapped configuration with a single logical ring is also a legal topology. If only one logical ring is present in an FDDI topology, it is designated as the primary ring. Multiple non-adjacent faults on the trunk ring produce multiple FDDI topologies, each with a disjoint logical ring.

An optional recovery process from a single fault, called Global Hold, may be employed. Global Hold is a policy whereby, if both logical rings are operational and a fault is detected in one of the logical rings, then the current configuration is held and all data traffic may be switched to the fault-free logical ring.

A single concentrator may connect one, or two in the case of a DAC, trees into a logical ring. This allows either the primary, the secondary, or both logical rings to be extended to include a tree.

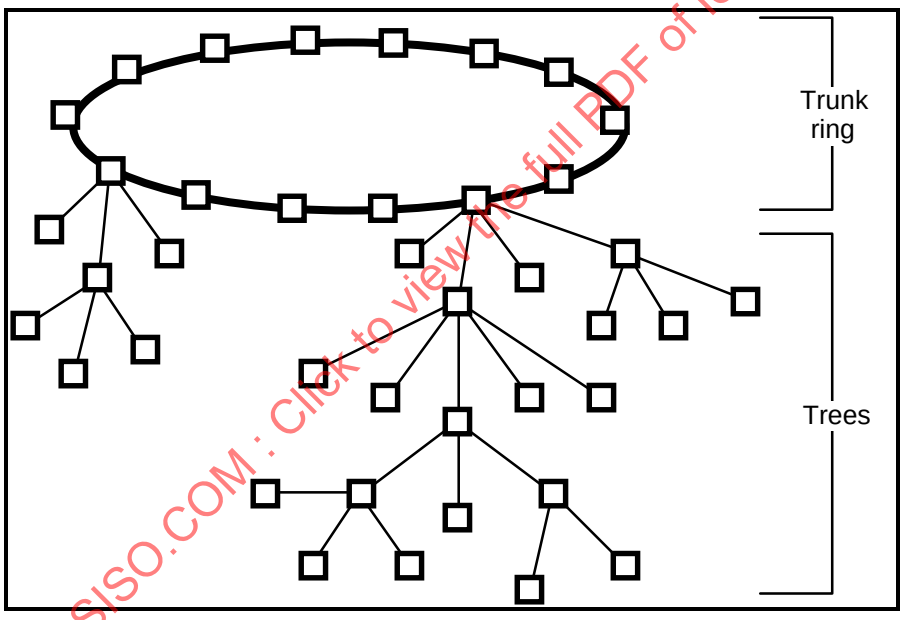


Figure 4 – Ring of trees topology

5.2.3 Physical media topology

The Physical Media topology may contain more than one dual ring. For example, additional connections may exist to provide redundancy within the concentrator trees. These redundant connections are not included in the token Path, which is limited to a single dual ring of trees, but are available as backup links. The construction of networks with redundant tree connections is allowed by ISO/IEC 9314, but the rules for efficient utilization of these connections are not described herein.

Legal FDDI topologies may be constructed with the FDDI Physical Media by following the connection rules set for Ports when connecting FDDI nodes together. This allows legal FDDI topologies to be constructed with the FDDI Physical Media.

5.2.4 FDDI connection rules

The types of Ports (A, B, M, or S) at both ends of a physical connection determine the characteristics of that physical connection. These characteristics include whether the connection will be accepted or rejected, whether SMT will be notified of potential connection problems, and the connection mode that will be established. Connections may be rejected to prevent the establishment of illegal or undesirable topologies. A connection may also be rejected by a neighbouring node because of that node's inability to support the connection. The connection rules matrix that follows summarizes the validity of and the action to be taken for each type of connection:

		Other Port			
		A	**B**	**S**	**M**
This Port	**A**	V,U	V	V,U	V,W
	B	V	V,U	V,U	V,W
	S	V,U	V,U	V	V
	M	V	V	V	I,U

where

V indicates a valid connection
I indicates an illegal connection
U indicates an undesirable connection with notification to SMT required
W indicates, if Active, prevent THRU in CFM and Port B takes precedence (with defaults)

The detailed connection rules for a specific Port (this Port) to other Ports are:

A to A — Undesirable peer connection that creates twisted primary and secondary rings, notify SMT.

A to B — Normal trunk ring peer connection.

A to S — Undesirable peer connection that creates a wrapped ring, notify SMT.

A to M — Tree connection with possible redundancy. Node shall not go to THRU state in CFM. My Port B shall have precedence (with defaults) for connecting to a Port M in a single MAC node.

B to A — Normal trunk ring peer connection.

B to B — Undesirable peer connection that creates twisted primary and secondary rings, notify SMT.

B to S — Undesirable peer connection that creates a wrapped ring, notify SMT.

B to M — Tree connection with possible redundancy. Node shall not go to THRU state in CFM. My Port B shall have precedence (with defaults) for connecting to a Port M in a single MAC node.

S to A — Undesirable peer connection that creates a wrapped ring, notify SMT.

S to B — Undesirable peer connection that creates a wrapped ring, notify SMT.

S to S — Connection that creates a single ring of two slave stations.

S to M — Normal tree connection.

M to A — Tree connection that provides possible redundancy.

M to B — Tree connection that provides possible redundancy.

M to S — Normal tree connection.

M to M — Illegal connection that creates a tree of rings topology.

When either Port A or Port B is connected to a Port M and the other Port is connected to a Port A, a Port B, or a Port S, then either connection may be withheld in a deterministic manner as specified herein (see clause 9). This avoids a dual attachment node attempting to form part of a tree at one of its Ports and part of a dual ring at the other Port.

Whether to accept or reject the connection depends upon the connection policies adopted by the nodes involved in the connection. The rule is to accept the connection if at least one node's policy is to allow such a connection, and to reject the connection if both nodes have a policy that disallows such a connection.

5.3 Overview of SMT functions

SMT is described herein in terms of various subcomponent functions. These are summarized, on a clause-by-clause basis, as follows.

Clause 6 specifies the services between SMT and other entities. Subclauses 6.1 to 6.3 specify the SMT services with the MAC, PHY, and PMD entities within the same node, whereas 6.4 specifies the services provided by SMT to System Management, and includes specification of the Management Information Base (MIB) of SMT.

Clause 7 specifies the frame formats and protocols used to manage stations in an FDDI ring on a peer-to-peer basis. Included, for instance, is the Neighbour Information Frame (NIF) that is used to periodically announce basic station description information, and the Status Information Frame (SIF) that is used to request and provide a station's operating and configuration information.

Also included in clause 7 are Status Report Frame (SRF) and Parameter Management Frame (PMF) types which are used in conjunction with the protocols specified in clause 8 to provide for remote management of nodes. Complete access to the MIB with read/write capabilities matching those locally available are specified.

Clause 9 specifies the Connection Management (CMT) for a node. CMT supports the wide variety of physical and logical topologies described earlier in this clause. CMT controls the establishment of a media attachment to the FDDI network, the connections with other nodes in the ring, and the internal configuration of the various entities within a node. CMT includes provision for a link confidence test and also specifies a Link Error Monitor (LEM) which monitors active links, on a link-by-link basis, to ensure that failing links are detected and, if required, removed from the network.

Clause 10 specifies the Ring Management (RMT) function which monitors MAC operation and takes actions necessary to aid in achieving an operational ring. RMT occurs on a per MAC basis and aids in the detection and resolution of failures, such as stuck beaconing and the presence of duplicate addresses that can have a global significance.

6 Services

This clause specifies the services provided by SMT and the services required by SMT. The services as defined herein do not imply any particular implementation, or any interface. Services described are

a) Services between the local MAC entity and SMT (indicated by SM_MA_ prefix);
b) Services between the local PHY entity and SMT (indicated by SM_PH_ prefix);
c) Services between the local PMD entity and SMT (indicated by SM_PM_ prefix);
d) SMT services provided to System Management.

6.1 SMT-to-MAC services

This clause specifies the services provided at the interface between the station management (SMT) entity and MAC. This interface is used by SMT to monitor and control the operation of a local MAC entity. Additional detail is provided in MAC concerning conditions that generate these primitives and MAC actions upon receipt of SMT generated primitives.

The following primitives are defined:

SM_MA_INITIALIZE_PROTOCOL.request
SM_MA_INITIALIZE_PROTOCOL.confirmation
SM_MA_CONTROL.request
SM_MA_STATUS.indication
SM_MA_UNITDATA.request
SM_MA_UNITDATA.indication
SM_MA_UNITDATA_STATUS.indication
SM_MA_TOKEN.request

Each primitive includes the information that is passed between the MAC and the SMT entities.

6.1.1 SM_MA_INITIALIZE_PROTOCOL.request

This primitive has local significance and is used by SMT to reset MAC and optionally change operational parameters of MAC.

Semantics of the primitive:

SM_MA_INITIALIZE_PROTOCOL.request (
individual_MAC_address (16 bit),
individual_MAC_address (48 bit),
group_MAC_addresses,
T_Min_value,
T_Max_value,
TVX_value,
T_Req_value,
T_Pri_value,
indicate_for_own_frame,
indicate_for_rcv_only_good_frame
)

All parameters of this primitive are optional. If a parameter is omitted, MAC shall use the most recently provided value for this parameter or, if no value has been previously provided, the default value for the parameter. References to the attributes contained in 6.4 for the individual parameters are provided in parentheses.

The individual_MAC_address is the octet string that MAC shall use as its individual address. Both a 16-bit address and a 48-bit address may be supplied (see 6.4 for MIB definitions).

Each group_MAC_address is an octet string that MAC shall use as a group address.

T_Min_value specifies the minimum target Token rotation time (TTRT) to be supported.

T_Max_value specifies the maximum target Token rotation time (TTRT) (ref. fddiMACT-Max).

TVX_value specifies the value of TVX to be used (ref. fddiMACTvxValue).

T_Req_value specifies the requested TTRT for asynchronous traffic (ref. fddiMACT-Req).

T_Pri_value specifies a set of priority Token rotation time thresholds (ref. fddiMACT-Pri0 – fddiMACT-Pri6).

The indicate_for_own_frame parameter specifies that MAC shall also receive and indicate to SMT frames that it has transmitted.

The indicate_for_rcv_only_good_frames parameter is the value MAC shall use to decide whether to generate MA_UNITDATA.indication and SM_MA_UNITDATA.indication primitives only on frames that are good, or alternatively on all frames.

When generated:

This primitive shall be generated by SMT whenever it requires MAC to reconfigure.

Effect upon receipt:

Receipt of this primitive causes MAC to establish the values of its addresses, timers, and other initialization parameters. Upon completion of this primitive, MAC generates a SM_MA_INITIALIZE_PROTOCOL.confirmation.

6.1.2 SM_MA_INITIALIZE_PROTOCOL.confirmation

This primitive is used by the MAC to inform SMT that the SM_MA_INITIALIZE_PROTOCOL.request is complete.

Semantics of the primitive:

```
SM_MA_INITIALIZE_PROTOCOL.confirmation    (
                                          status
                                          )
```

The status parameter indicates the success or failure of the SM_MA_INITIALIZE_PROTOCOL.request.

When generated:

This primitive is generated by MAC upon completion of a SM_MA_INITIALIZE_PROTOCOL.request.

Effect upon receipt:

The receipt of this primitive by SMT shall signify to SMT that the desired reconfiguration has been accomplished.

6.1.3 SM_MA_CONTROL.request

This primitive has local significance and is used by SMT to control the operation of a local MAC entity.

Semantics of the primitive:

```
SM_MA_CONTROL.request     (
                          control_action,
                          beacon_information,
                          requested_status,
                          requested_condition
                          )
```

The control_action parameter shall include the following: Reset, Beacon, Present_Status, Reset_Counters, interrupt_upon_condition, send bad FCS, or optionally Claim.

The beacon_information parameter shall contain the DA field and the Information field for use in the Beacon Frame if the control_action parameter specifies Beacon.

The requested_status parameter shall include the following: current values of all counters, current values of all timers, current receiver state machine state, and current transmitter state machine state.

The requested_condition parameter shall include: capture of Token, receipt of frame.

When generated:

This primitive is generated by SMT to cause MAC to take the action specified by the control_action parameter.

Effect upon receipt:

Receipt of this primitive by MAC causes it to take the following actions:

a) If the control_action parameter is Reset or Beacon then MAC shall:

generate the MAC_Reset signal

enter receiver state R0

enter transmitter state T0;

b) If the control_action parameter is Beacon, then following the above, State T5 shall be entered in the transmitter state machine. The Beacon Frames shall be constructed using the beacon_information parameter;

c) If the control_action parameter is Present_Status, then MAC shall present the status to SMT as indicated by the requested_status parameter;

d) If the control_action parameter is Reset_Counters, then MAC shall reset all counters;

e) If the control_action parameter is interrupt_upon_condition, then MAC shall signal SMT when the requested_condition is detected;

f) If the control_action parameter is send bad FCS, then MAC shall send bad FCS with a specific frame to be transmitted.

6.1.4 SM_MA_STATUS.indication

This primitive is used by MAC to inform SMT of errors and significant status changes.

Semantics of the primitive:

SM_MA_STATUS.indication (status_report)

The status_report parameter specifies appropriate status including the following:

a) any change of Ring_Operational status;

b) receipt of My_Claim, Higher_Claim or Lower_Claim;

c) receipt of My_Beacon or Other_Beacon;

d) expiration of TVX;

e) expiration of TRT and Late_Ct _ 0;

f) receipt of PH_Invalid causing the receiver to enter State R0;

g) receipt of MAC_Reset causing the receiver to enter State R0;

h) overflow of Frame_Ct, Error_Ct or Lost_Ct;

i) receipt of a frame with the Destination Address equal to this station's individual address and the received A Indicator set;

j) recognition that this station's transmitted frame was lost in the course of its circulation around the ring;

k) anytime the R_Flag goes from reset to set;

l) anytime a station sets Ax but does not set Cx;

m) receipt of a Token if the MAC transmitter is in State T2 or T3;

n) expiration of TRT in MAC transmitter State T4 or T5;

o) occurrence of any condition specified in a previously received SM_MA_CONTROL.request (interrupt_upon_condition);

When generated:

This primitive is generated by MAC when any of the listed reportable conditions are detected.

Effect upon receipt:

The receipt of this primitive shall cause SMT to take the actions specified in this International Standard.

6.1.5 SM_MA_UNITDATA.request

This primitive defines the transfer of one or more SMT Service Data Units (SDU) from SMT. Any defined PDU format may be sent.

Semantics of the primitive:

SM_MA_UNITDATA.request (
FC_value (1),
destination_address (1),
M_SDU (1),
requested_service_class (1),
stream (1),
FC_value (2),
destination_address (2),
M_SDU (2),
requested_service_class (2),
stream (2),
•
•
•
FC_value (n),
destination_address (n),
M_SDU (n),
requested_service_class (n),
stream (n),
Token_class
)

Each set of FC_value, destination_address, M_SDU, requested_service_class and stream parameters specifies one frame for transmission and is referred to as a subrequest.

The FC_value parameter supplies the FC field to be transmitted as part of the frame.

The destination_address parameter may specify either an individual or a group MAC address. It shall contain sufficient information to create the DA field that is appended to the frame by MAC. Address length is determined by the L bit of the associated FC_value parameter.

Each M_SDU parameter specifies an SMT service data unit as received at the MAC interface to be transmitted by MAC. There is sufficient information associated with the M_SDU for MAC to determine the length of the service data unit. Associated with each M_SDU is a requested_service_class parameter.

Requested_service_class may be either synchronous, asynchronous or immediate. If asynchronous the priority level may also be specified. If not specified the default value is assumed. Immediate, if specified, shall cause the frame to be sent immediately without waiting for receipt of a Token.

NOTE 1 -The use of synchronous as the requested_service_class is dependent upon allocation of the required synchronous bandwidth.

Stream is a parameter which, if set, shall cause multiple M_SDUs to be transmitted as a result of the same SM_MA_UNITDATA.request. Stream, when reset, indicates that this SDU is the last one associated with this SM_MA_UNITDATA.request. The frames shall be transmitted in the order presented by this primitive regardless of the associated requested_service_class. If TRT has expired (Late_Ct not= 0) or if a frame is encountered which cannot be transmitted because of its associated requested_service_class and the current value of THT, then transmission is terminated and a Token is issued as defined by the Token_class parameter. A SM_MA_UNITDATA_STATUS.indication is subsequently returned to SMT. If the transmission_status is successful, then MAC may initiate transmission of the remaining frames on the next permitted access opportunity or alternatively, MAC may require a new SM_MA_UNITDATA.request.

Token_class specifies the class of Token MAC shall issue following the transmission of the associated SDUs if no other request that can be honoured is pending. It may specify either none, restricted or non-restricted. If no SDUs were specified by the SM_MA_UNITDATA.request, then MAC shall issue the requested class of Token.

When generated:

This primitive shall be generated by SMT whenever data is to be transferred to a peer SMT entity or entities or a Token is to be generated.

Effect of Receipt:

The receipt of this primitive causes MAC to append all MAC specific fields, including DA, SA, and any fields that are unique to the medium access method, and pass the properly formed frames to the lower layers of protocol for transfer to the peer MAC entity or entities.

NOTE 2 – The capture of a Token is implicit in this primitive and therefore it is not necessary to issue an SM_MA_TOKEN.request primitive in conjunction with it.

6.1.6 SM_MA_UNITDATA.indication

This primitive defines the transfer of data from a local MAC entity to SMT. This primitive shall be able to report any SMT or MAC frame addressed to the station.

Semantics of the primitive:

```
SM_MA_UNITDATA.indication    (
                             FC_value,
                             destination_address,
                             source_address,
                             M_SDU,
                             reception_status
                             )
```

The FC_value parameter specifies the value for the frame's FC field. The destination_address parameters may be either an individual or a group address as specified by the DA field of the incoming frame. The source_address parameter is an individual address as specified by the SA field of the incoming frame. The M_SDU parameter shall specify the MAC service data unit as received by the local MAC entity.

The reception_status parameter indicates the success or failure of the incoming frame. It consists of the following elements:

a) frame validity: FR_GOOD, FR_BAD

If a FR_BAD is reported, the reason for the error shall also be reported. The reason shall be one of the following:

1) Invalid FCS – calculated FCS does not match the received FCS

2) Length error – the frame did not have a valid data length

3) Internal error – an internal error has occurred which prevents MAC from transferring to SMT a frame that has been acknowledged by the setting of the A and C indicators;

b) frame status:

The received E, A, C, and, optionally, any other indicator values.

When generated:

The SM_MA_UNITDATA.indication primitive shall be generated by MAC to indicate the arrival of a MAC or SMT frame addressed to this station at the local MAC entity. If so initialized (ref. SM_MA_INITIALIZE.PROTOCOL.request), the MAC shall also generate this primitive upon receipt of frames that it has transmitted. This allows examination of elements of the reception_status for a transmitted frame following circulation around the ring.

Effect of Receipt:

The receipt of this primitive shall cause SMT to take the actions specified in this International Standard.

6.1.7 SM_MA_UNITDATA_STATUS.indication

This primitive shall provide an appropriate response to the SM_MA_UNITDATA.request primitive generated by SMT signifying the success or failure of the request.

Semantics of the primitive:

```
SM_MA_UNITDATA_STATUS.indication    (
                                    number_of_SDUs,
                                    transmission_status,
                                    provided_service_class
                                    )
```

The number_of_SDUs parameter reports the number of M_SDUs transmitted on a given access opportunity as a result of this request.

The transmission_status parameter shall be used to pass information back to the local requesting SMT entity. It shall be used to indicate the success or failure of the previous associated SM_MA_UNITDATA.request. If the SM_MA_UNITDATA.request primitive specified more than one M_SDU then the transmission_status parameter shall apply to all of the SDUs transmitted indicating if all were acknowledged, via the A and C indicators, by a peer MAC entity.

The provided_service_class parameter specifies the service class that was provided for the transfer.

When generated:

This primitive is generated by the MAC entity in response to an SM_MA_UNITDATA.request primitive from SMT.

Effect of Receipt:

The receipt of this primitive by SMT shall indicate to SMT the success or failure of the request.

NOTE 3 – In the event of multiple outstanding requests, additional information may be required in order for SMT to associate the response with the appropriate request. The association may be implied by the requests being serviced in a first-in-first-out manner. Alternatively, MAC may maintain multiple queues of requests, at least one for each class_of_service implemented, servicing each of these queues in a first-in-first-out manner. It is assumed that if the addressed peer MAC entity, or entities, acknowledge receipt of the frame(s) (by setting the A and C indicators), then either the frame(s) will be delivered, or an internal error will be reported, to the corresponding SMT entity, or entities, via an SM_MA_UNITDATA.indication.

6.1.8 SM_MA_TOKEN.request

This primitive is used by SMT to request the capture of the next Token.

Semantics of the primitive:

```
SM_MA_TOKEN.request    (
                       requested_Token_class
                       )
```

Requested_Token_class may be either restricted or non-restricted. The priority level may also be specified if multiple levels of priorities are implemented.

When generated:

This primitive may be generated by the local SMT entity when data of a time critical nature is to be transferred.

Effect of Receipt:

The receipt of this primitive causes MAC to capture the next usable Token based on the requested_Token_class parameter. MAC then enters State T2 and transmits Idle symbols until an SM_MA_UNITDATA.request primitive is received from SMT unless TRT expires first, in which case MAC shall issue another Token of the same Token_class as was captured.

NOTE 4 – This primitive may be used for time critical operations to minimize the effects of ring latency. This mode of operation may cause longer than usual preambles preceding a frame and may lock out other stations from gaining fair access to transmit data. It is an additional mode of operation which results in the capture of an appropriate Token following the receipt of SM_MA_UNITDATA.request and therefore should not be used for non-time critical transfers of data on the FDDI ring.

6.2 SMT-to-PHY services

The services supplied by PHY allow SMT to control the operation of PHY. Additional detail is provided in PHY concerning conditions that generate these primitives and PHY actions upon receipt of SMT generated primitives. The following primitives are defined:

SM_PH_LINE-STATE.request
SM_PH_STATUS.indication
SM_PH_CONTROL.request

All primitives described in this subclause are mandatory. Each primitive includes the information that will be passed between PHY and SMT.

6.2.1 SM_PH_LINE-STATE.request

This primitive is generated by SMT to request PHY to send a continuous stream of similar symbols.

Semantics of the primitive:

SM_PH_Line-State.request (Line-State_action)

The Line-State_action parameter shall be one of the following:

TRANSMIT_QUIET – This parameter is asserted to cause PHY to send a continuous stream of Quiet symbols. In this condition the medium transmitter generates no transitions.

NOTE 5 – It is recommended that the transmitter be inactive (i.e. no light output).

TRANSMIT_HALT – This parameter is asserted to cause PHY to send a continuous stream of Halt symbols.

TRANSMIT_IDLE – This parameter is asserted to cause PHY to send a continuous stream of Idle symbols.

TRANSMIT_MASTER – This parameter is asserted to cause PHY to send a continuous stream of alternating Halt and Quiet symbol pairs.

TRANSMIT_PDR – This parameter is asserted to cause PHY to send the stream of symbols presented to the PH_UNITDATA.request interface.

When generated:

These primitives shall be generated by SMT as part of station insertion or removal sequences.

Effect upon receipt:

PHY sends a continuous stream of the commanded symbol or symbol pair out the outbound medium.

6.2.2 SM_PH_STATUS.indication

This primitive is generated by PHY to inform SMT of Line-State activity and status changes. The specific items reported are defined in the following.

Semantics of the primitive:

```
SM_PH_STATUS.indication     (
                           status_report
                           )
```

The status_report parameter shall be one of the following:

QUIET_LINE-STATE_RECEIVED (QLS) – This parameter is asserted by PHY when the Quiet_Line-State is received on the inbound medium.

HALT_LINE-STATE_RECEIVED (HLS) – This parameter is asserted by PHY when the Halt_Line-State is received on the inbound medium.

MASTER_LINE-STATE_RECEIVED (MLS) – This parameter is asserted by PHY when the Master_Line-State is received on the inbound medium.

IDLE_LINE-STATE_RECEIVED (ILS) – This parameter is asserted by PHY when the Idle_Line-State is received on the inbound medium.

ACTIVE_LINE-STATE_RECEIVED – This parameter is asserted by PHY when the Active_Line-State (ALS) is entered.

NOISE_LINE-STATE_RECEIVED – This parameter is asserted by PHY when the Noise_Line-State (NLS) is entered.

LINE-STATE_UNKNOWN – This parameter is asserted by PHY when any of the defined line states are exited and the entry conditions to a new line state have not yet been satisfied. Also included is an indication of the most recently known line state.

When generated:

These primitives are generated by PHY to signal the occurrence of the indicated condition.

Effect upon receipt:

The receipt of this primitive shall cause SMT to take the actions specified in the state machines contained herein.

6.2.3 SM_PH_CONTROL.request

This primitive has local significance and is used by SMT to control the operation of PHY.

Semantics of the primitive:

```
SM_PH_CONTROL.request      (
                           Control_Action,
                           Requested_Status
                           )
```

The Control_Action parameter shall include the following: Reset, Present_Status, Begin_Loopback, or Cancel_Loopback.

The Requested_Status parameter causes PHY to report its status which includes the current line state. If the current line state is unknown, then LINE-STATE_UNKNOWN is reported along with the most recent known line state.

When generated:

This primitive shall be generated by SMT to cause PHY to take the action specified by the Control_Action parameter.

Effect upon receipt:

PHY takes the requested action.

6.3 SMT-to-PMD services

The services supplied by PMD allow SMT to control the operation of PMD. The PMD performs the requested SMT services preemptively over any requested PHY services. Additional detail is provided in PMD concerning conditions that generate these primitives and PMD actions upon receipt of SMT generated primitives.

The following primitives are defined:

SM_PM_CONTROL.request
SM_PM_BYPASS.request
SM_PM_SIGNAL.indication

Each primitive includes the information that is passed between the PMD and SMT entities.

6.3.1 SM_PM_CONTROL.request

This primitive is generated by SMT and asserted to PMD to force the transmit function to place a logic "0" optical signal on the outbound medium.

Semantics of the primitive:

SM_PM_CONTROL.request (Control_Action)

The Control_Action parameter shall include the following: Transmit_Enable and Transmit_Disable.

When generated:

SMT shall generate this primitive whenever it wants to enable or disable the PMD optical transmitter.

Effect upon receipt:

Receipt of this primitive by PMD with a Control_Action parameter of Transmit_Disable causes PMD to transmit a logic "0"optical signal (i.e. low light) preemptively over the PM_UNITDATA.request primitive.

Receipt of this primitive by PMD with a Control_Action parameter of Transmit_Enable causes PMD to transmit the optical signal requested by the PM_UNITDATA.request primitive.

6.3.2 SM_PM_BYPASS.request

This primitive is generated by SMT and asserted to PMD to indicate that SMT wants to join or leave the FDDI network.

Semantics of the primitive:

SM_PM_BYPASS.request (Control_Action)

The Control_Action parameter shall include the following: Insert, Deinsert.

When generated:

SMT shall generate this primitive whenever it wants to activate or deactivate the optical switch(es).

Effect upon receipt:

Upon receipt of this primitive with a Control_Action parameter of Insert, PMD activates the optical switch such that the MIC inbound optical signal from the cable plant is directed to the optical receiver. The output of the optical transmitter shall be directed to the MIC output to the cable plant.

Upon receipt of this primitive with a Control_Action parameter of Deinsert, PMD deactivates the optical switch such that the MIC inbound optical signal from the cable plant is directed through the switch to the MIC output to the cable plant. The output of the optical transmitter shall be directed through the optical switch to the input of the optical receiver. This state is called the bypassed mode.

NOTE 6 – Optical bypass switches are optional in an FDDI network. Stations that do not employ optical switches do not require this service.

6.3.3 SM_PM_SIGNAL.indication

This primitive is generated by PMD and asserted to SMT to indicate the status of the optical signal level being received by PMD.

Semantics of the primitive:

```
SM_PM_SIGNAL.indication        (
                               Signal_Detect(status)
                               )
```

The Signal_Detect(status) parameter indicates if the inbound optical signal level is above (status = on) or below (status = off) the optical signal detection threshold.

When generated:

PMD generates this primitive to indicate the status of the Signal_Detect.

Effect upon receipt:

SMT shall use this primitive to determine the status of the optical signal level being received by PMD and shall take actions as specified in the state machines contained herein.

6.4 SMT services to systems management

This subclause provides the specification of management information related to the operation of the SMT. SMT management information is defined by specifying:

- the syntax and semantics of the information elements;
- the relationship of the information elements to the operation of SMT and to other information elements of SMT;
- the operations on the information elements that are available to Systems Management.

6.4.1 Overview of SMT management services

The SMT management information that is supplied to systems management is represented using an object oriented approach consistent with the approach taken by OSI Management Standards. See ISO/IEC 7498-4:1989 and ISO/IEC 10165-4:1992. Station Management Information is described in terms of Managed Objects. Figure 5 depicts the SMT Object Model. It shows the relationship of managed objects within an FDDI station and the components of SMT: the SMT frame services (see clause 8), Connection Management (see clause 9), and Ring Management (see clause 10).

This part of ISO/IEC 9314 defines four Managed Object Classes. The SMT managed object class models the management information for an FDDI station (node). The MAC managed object class models the management information required for the MAC entities within a station. The Path managed object class models the management information required for the management of configuration Paths within a station. The Port object class models the management information required for the PHY/PMD entities within a station.

A particular instance of a managed object class is called a managed object. A particular FDDI station can have multiple MAC, Path, and Port managed objects but only one SMT managed object. The management of these managed objects is accomplished through notifications and operations.

Notifications are unsolicited reports generated by a managed object. Operations are functions performed by a management agent process on a managed object. Get and Set operations are defined in this part of ISO/IEC 9314.

6.4.2 SMT-Management agent process local service primitives

The definition of the local primitives or service interfaces, that are used to read and write attributes, or to cause an action or report an event, is beyond the scope of this part of ISO/IEC 9314.

NOTE 7 – SMT uses the MSB address form in its Management Information Base. Some standard management agent processes use the Canonical representation for interchange of addresses. The implementer should be aware that in this case, conversion between MSB representation and Canonical representation is required when the management agent process accesses a station's MIB either locally, or remotely through SMT frames.

6.4.3 Management information base (MIB) structure

The SMT managed object classes are defined using the formats defined in the ISO 10165-4, hereafter referred to as GDMO.

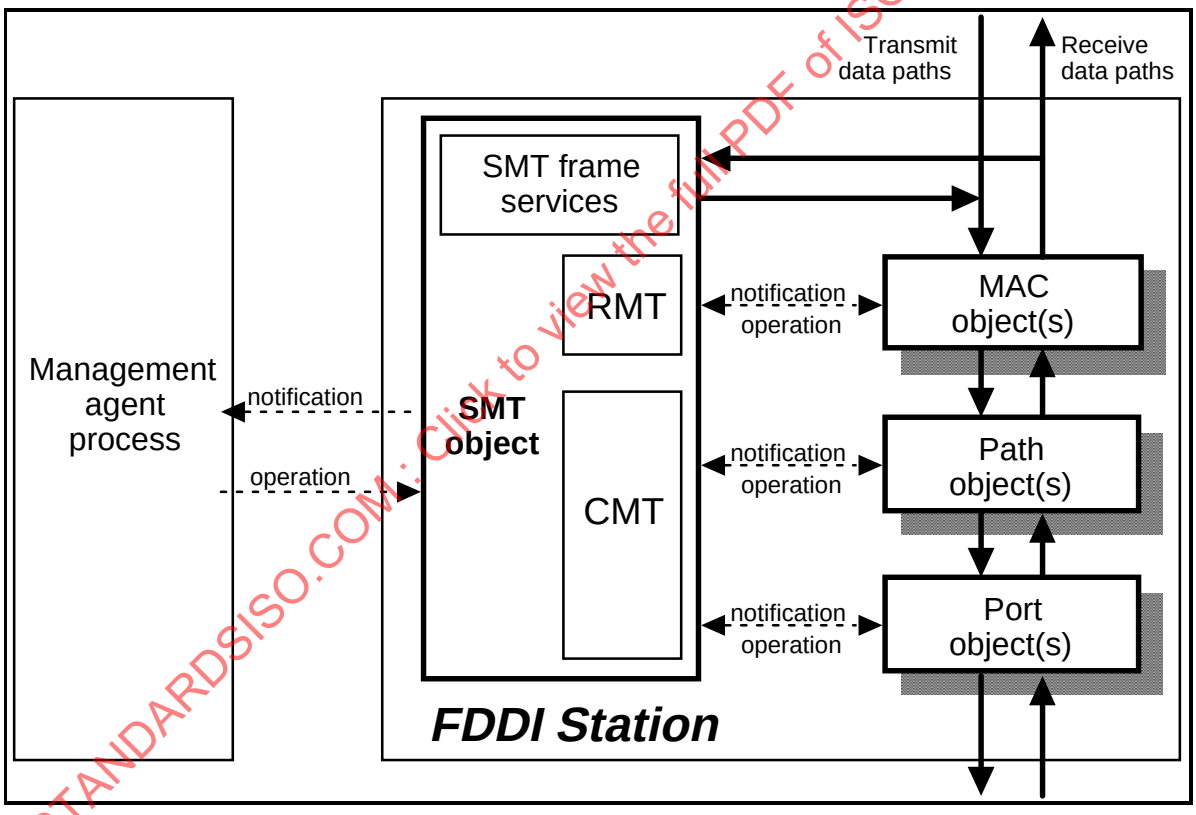


Figure 5 – SMT management model

GDMO presents specific structures, called templates, in a formal description language, that provides the information necessary for the application of object oriented design principles in the definition of OSI management for Open Systems, for protocol layers, and for communication components.

These templates serve to describe the managed object classes in a precise manner. Each managed object class is defined using templates that describe the behaviour, attributes, notifications, and actions for that class.

The data structures required to support the GDMO templates are defined using ISO 8824. ASN.1 is a formal notation for documenting data structures. The templates for the four SMT-defined managed object classes are contained in the management information definitions (see 6.4.5). This part of ISO/IEC 9314 uses a template format that places GDMO templates within templates. This use of "inline templates" improves the readability of this part of ISO/IEC 9314. A summary of the characteristics of the object classes is found in the MIB Summary (see 6.4.5.1).

The managed object class templates (see 6.4.5.2) contain the definition of the object classes. This definition includes the specification of the following characteristics.

a) Object class behaviour;

b) Mandatory object class attributes;

c) Mandatory object class notifications;

d) Mandatory object class actions;

e) Conditional object class attributes;

f) Conditional object class notifications;

g) Conditional object class actions.

The conditional aspects of a managed object are represented in the object class template as a conditional package. The conditional packages define the conditions under which conditional attributes, notifications, and actions are included in an object instance.

The attribute group templates (see 6.4.5.3) contain the definitions of attribute groups within an object class. These templates specify the attributes that are operated on when group operations are supported by a station. This part of ISO/IEC 9314 specifies the PMF GET as the only group operation.

The attribute templates (see 6.4.5.4) define the attributes of an object class. These templates define the behaviour and registration of a particular attribute. The "WITH ATTRIBUTE SYNTAX" clause indicates the ASN.1 definition of the attribute.

The action templates (see 6.4.5.5) define the actions of an object class. These templates define the behaviour, confirmation mode, and registration of a particular action. The "WITH INFORMATION SYNTAX" clause indicates the ASN.1 definition of the action. The "WITH REPLY SYNTAX" clause indicates the ASN.1 definition of the reply to the action.

The notification templates (see 6.4.5.6) define the notifications of an object class. These templates define the behaviour, confirmation mode, and registration of a particular notification. The "WITH INFORMATION SYNTAX" clause indicates the ASN.1 definition of the notification. All notifications in this part of ISO/IEC 9314 are unconfirmed.

The ASN.1 definitions (see 6.4.5.7) contain the formal representation of the data structure of each attribute, action, and notification.

The name binding (see 6.4.5.8) contains a formal description of the naming scheme associated with the object classes. The name binding templates indicate how instances of managed object classes are named within a station.

6.4.4 Integrity of MIB state

The MAC, Path and Port objects may have multiple instances within the MIB. In order to support implementations with flexible internal configurations, a MAC or Port object instance in the MIB may not necessarily be associated with a physical resource of that type within the station. Each MAC and Port object instance and its associated attributes are identified by a resource index within the object type. These indexes, assigned by the implementer, shall be non-zero.

An implementation is not required to maintain the state of any MIB object across discontinuities in the management of that object. Permitted discontinuities include SMT initialization, power off/on transitions, self-test, and, for MAC and Port objects, unavailable/available transitions of the fddiMACHardwarePresent or fddiPORTHardwarePresent attributes. It is also possible that detection of a major inconsistency in the internal state of SMT may cause a discontinuity.

An implementation is not required to maintain the state of any object across discontinuities in the instantiation of that object. The mandatory package shall be included in all instantiations of that object class. Any discontinuity shall cause the object to be re-instantiated. A conditional package shall be included in an instantiation of an object class when the conditions for its presence are satisfied.

After any discontinuity in the instantiation of an object class its state components (attributes, actions, and notifications) shall be deterministically set to their initial values, with the following exceptions.

a) A resource index attribute may assume any prior value, if known; otherwise it shall be assigned an initial value. There may be less impact to management applications if a prior value is used.

b) A statistical counter attribute may assume its latest prior value, if known, and if the resource index for its associated object is known not to have changed; otherwise it shall be reset to its initial value. The value of statistical information is generally enhanced if the continuity of counters is maintained.

c) A time stamp or set count attribute may assume any value greater than or equal to its latest prior value, if known, and if the StationId is known not to have changed; otherwise it shall be reset to its initial value. The value of statistical information is generally enhanced if, when the continuity of counters is maintained, the continuity of time stamps is also maintained.

When a request to alter a MIB attribute is honoured, either from the local systems management entity or via the Parameter Management protocol, SMT shall set the value of the attribute exactly to the requested value.

When a MIB attribute is altered or a MIB action is invoked, either from the local systems management entity or via the Parameter Management protocol, SMT shall ascertain whether the resulting MIB state is at variance with the current operation of the node. If there is a variance, SMT shall act to restore consistency in a timely and deterministic manner.

For those attributes designated read-write for which a specific range of values is defined in the MIB, the entire range of values shall be supported, unless otherwise specified in the attribute behaviour description.

6.4.5 Management information definitions

6.4.5.1 MIB summary

The following sections summarize the FDDI Management Information Base. Each group, attribute, action and notification is listed by its name, access (G = GET, GR = GET-REPLACE, S = PMF-Set, NA = Not Accessible), status (M = MANDATORY, MP = MANDATORY if hardware present, O = OPTIONAL, OP = OPTIONAL but present only when hardware present), Parameter_Type, and abbreviated registration information. This summary is for information purposes only; if any differences exist, the formal definitions have precedence.

6.4.5.1.1 SMT attributes

fddiSMTStationIdGrp	**G**	**M**	**10 0A**	**fddiSMT 10**
fddiSMTStationId	G	M	10 0B	fddiSMT 11
fddiSMTOpVersionId	G	M	10 0D	fddiSMT 13
fddiSMTHiVersionId	G	M	10 0E	fddiSMT 14
fddiSMTLoVersionId	G	M	10 0F	fddiSMT 15
fddiSMTManufacturerData	G	O	10 10	fddiSMT 16
fddiSMTUserData	GR	M	10 11	fddiSMT 17
fddiSMTMIBVersionId	G	M	10 12	fddiSMT 18
fddiSMTStationConfigGrp	**G**	**M**	**10 14**	**fddiSMT 20**
fddiSMTMac-Ct	G	M	10 15	fddiSMT 21
fddiSMTNonMaster-Ct	G	M	10 16	fddiSMT 22
fddiSMTMaster-Ct	G	M	10 17	fddiSMT 23
fddiSMTAvailablePaths	G	M	10 18	fddiSMT 24
fddiSMTConfigCapabilities	G	M	10 19	fddiSMT 25
fddiSMTConfigPolicy	GR	M	10 1A	fddiSMT 26

fddiSMTConnectionPolicy	GR	M	10 1B	fddiSMT 27
fddiSMTT-Notify	GR	M	10 1D	fddiSMT 29
fddiSMTStatRptPolicy	GR	M	10 1E	fddiSMT 30
fddiSMTTrace-MaxExpiration	GR	M	10 1F	fddiSMT 31
fddiSMTPORTIndexes	G	M	10 20	fddiSMT 32
fddiSMTMACIndexes	G	M	10 21	fddiSMT 33
fddiSMTBypassPresent	G	M	10 22	fddiSMT 34
fddiSMTStatusGrp	**G**	**M**	**10 28**	**fddiSMT 40**
fddiSMTECMState	G	M	10 29	fddiSMT 41
fddiSMTCF-State	G	M	10 2A	fddiSMT 42
fddiSMTHoldState	G	O	10 2B	fddiSMT 43
fddiSMTRemoteDisconnectFlag	G	M	10 2C	fddiSMT 44
fddiSMTStationStatus	G	M	10 2D	fddiSMT 45
fddiSMTPeerWrapFlag	G	M	10 2E	fddiSMT 46
fddiSMTMIBOperationGrp	**G**	**M**	**10 32**	**fddiSMT 50**
fddiSMTTimeStamp	G	M	10 33	fddiSMT 51
fddiSMTTransitionTimeStamp	G	M	10 34	fddiSMT 52
fddiSMTSetCount	G	O	10 35	fddiSMT 53
fddiSMTLastSetStationId	G	O	10 36	fddiSMT 54
Not Grouped				
fddiSMTVendorAttrib	GR	O	10 FF	fddiSMT 255

6.4.5.1.2 MAC attributes

fddiMACCapabilitiesGrp	**G**	**MP**	**20 0A**	**fddiMAC 10**
fddiMACFrameStatusFunctions	G	MP	20 0B	fddiMAC 11
fddiMACBridgeFunctions	G	OP	20 0C	fddiMAC 12
fddiMACT-MaxCapability	G	MP	20 0D	fddiMAC 13
fddiMACTVXCapability	G	MP	20 0E	fddiMAC 14
fddiMACConfigGrp	**G**	**M**	**20 14**	**fddiMAC 20**
fddiMACAvailablePaths	G	MP	20 16	fddiMAC 22
fddiMACCurrentPath	G	MP	20 17	fddiMAC 23
fddiMACUpstreamNbr	G	MP	20 18	fddiMAC 24
fddiMACDownstreamNbr	G	MP	20 19	fddiMAC 25
fddiMACOldUpstreamNbr	G	MP	20 1A	fddiMAC 26
fddiMACOldDownstreamNbr	G	MP	20 1B	fddiMAC 27
fddiMACDupAddressTest	G	MP	20 1D	fddiMAC 29
fddiMACRequestedPaths	GR	M	20 20	fddiMAC 32
fddiMACDownstreamPORTType	G	MP	20 21	fddiMAC 33
fddiMACIndex	G	M	20 22	fddiMAC 34
fddiMACAddressGrp	**G**	**MP**	**20 28**	**fddiMAC 40**
fddiMACSMTAddress	G	MP	20 29	fddiMAC 41
fddiMACLongGrpAddress	G	OP	20 2C	fddiMAC 44
fddiMACShortGrpAddress	G	OP	20 2D	fddiMAC 45
fddiMACOperationGrp	**G**	**MP**	**20 32**	**fddiMAC 50**
fddiMACT-Req	G	MP	20 33	fddiMAC 51
fddiMACT-Neg	G	MP	20 34	fddiMAC 52
fddiMACT-Max	G	MP	20 35	fddiMAC 53
fddiMACTvxValue	G	MP	20 36	fddiMAC 54
fddiMACT-Pri0	G	OP	20 38	fddiMAC 56
fddiMACT-Pri1	G	OP	20 39	fddiMAC 57
fddiMACT-Pri2	G	OP	20 3A	fddiMAC 58
fddiMACT-Pri3	G	OP	20 3B	fddiMAC 59
fddiMACT-Pri4	G	OP	20 3C	fddiMAC 60
fddiMACT-Pri5	G	OP	20 3D	fddiMAC 61
fddiMACT-Pri6	G	OP	20 3E	fddiMAC 62

fddiMACCountersGrp	**G**	**MP**	**20 46**	**fddiMAC 70**
fddiMACFrame-Ct	G	MP	20 47	fddiMAC 71
fddiMACCopied-Ct	G	MP	20 48	fddiMAC 72
fddiMACTransmit-Ct	G	MP	20 49	fddiMAC 73
fddiMACToken-Ct	G	OP	20 4A	fddiMAC 74
fddiMACError-Ct	G	MP	20 51	fddiMAC 81
fddiMACLost-Ct	G	MP	20 52	fddiMAC 82
fddiMACTvxExpired-Ct	G	OP	20 53	fddiMAC 83
fddiMACNotCopied-Ct	G	OP	20 54	fddiMAC 84
fddiMACLate-Ct	G	OP	20 55	fddiMAC 85
fddiMACRingOp-Ct	G	OP	20 56	fddiMAC 86
fddiMACFrameErrorConditionGrp	**G**	**M**	**20 5A**	**fddiMAC 90**
fddiMACFrameErrorThreshold	GR	M	20 5F	fddiMAC 95
fddiMACFrameErrorRatio	G	MP	20 60	fddiMAC 96
fddiMACNotCopiedConditionGrp	**G**	**O**	**20 64**	**fddiMAC 100**
fddiMACNotCopiedThreshold	GR	O	20 67	fddiMAC 103
fddiMACNotCopiedRatio	G	OP	20 69	fddiMAC 105
fddiMACStatusGrp	**G**	**M**	**20 6E**	**fddiMAC 110**
fddiMACRMTState	G	MP	20 6F	fddiMAC 111
fddiMACDA-Flag	G	MP	20 70	fddiMAC 112
fddiMACUNDA-Flag	G	MP	20 71	fddiMAC 113
fddiMACFrameErrorFlag	G	MP	20 72	fddiMAC 114
fddiMACNotCopiedFlag	G	OP	20 73	fddiMAC 115
fddiMACMA-UnitdataAvailable	G	MP	20 74	fddiMAC 116
fddiMACHardwarePresent	G	M	20 75	fddiMAC 117
fddiMACMA-UnitdataEnable	GR	M	20 76	fddiMAC 118
Not Grouped				
fddiMACVendorAttrib	GR	O	20 FF	fddiMAC 255

6.4.5.1.3 PATH attributes

fddiPATHConfigGrp	**G**	**M**	**32 0A**	**fddiPATH 10**
fddiPATHIndex	G	M	32 0B	fddiPATH 11
fddiPATHRingLatency	GR	O	32 0D	fddiPATH 13
fddiPATHTraceStatus	G	O	32 0E	fddiPATH 14
fddiPATHSbaPayload	GR	O	32 0F	fddiPATH 15
fddiPATHSbaOverhead	GR	O	32 10	fddiPATH 16
fddiPATHConfiguration	G	M	32 12	fddiPATH 18
fddiPATHT-Rmode	GR	O	32 13	fddiPATH 19
fddiPATHSbaAvailable	GR	O	32 14	fddiPATH 20
fddiPATHTVXLowerBound	GR	M	32 15	fddiPATH 21
fddiPATHT-MaxLowerBound	GR	M	32 16	fddiPATH 22
fddiPATHMaxT-Req	GR	M	32 17	fddiPATH 23
Not Grouped				
fddiPATHVendorAttrib	GR	O	32 FF	fddiPATH 255

6.4.5.1.4 PORT attributes

fddiPORTConfigGrp	**G**	**M**	**40 0A**	**fddiPORT 10**
fddiPORTMy-Type	G	M	40 0C	fddiPORT 12
fddiPORTNeighbourType	G	MP	40 0D	fddiPORT 13
fddiPORTConnectionPolicies	GR	M	40 0E	fddiPORT 14
fddiPORTMACIndicated	G	MP	40 0F	fddiPORT 15
fddiPORTCurrentPath	G	MP	40 10	fddiPORT 16
fddiPORTRequestedPaths	GR	M	40 11	fddiPORT 17
fddiPORTMACPlacement	G	MP	40 12	fddiPORT 18
fddiPORTAvailablePaths	G	MP	40 13	fddiPORT 19

fddiPORTMACLoop-Time	GR	O	40 15	fddiPORT 21
fddiPORTPMDClass	G	MP	40 16	fddiPORT 22
fddiPORTConnectionCapabilities	G	MP	40 17	fddiPORT 23
fddiPORTIndex	G	M	40 1D	fddiPORT 29
fddiPORTOperationGrp	**G**	**MP**	**40 1E**	**fddiPORT 30**
fddiPORTMaint-LS	GR	O	40 1F	fddiPORT 31
fddiPORTBS-Flag	G	MP	40 21	fddiPORT 33
fddiPORTPC-LS	G	OP	40 22	fddiPORT 34
fddiPORTErrorCtrsGrp	**G**	**MP**	**40 28**	**fddiPORT 40**
fddiPORTEBError-Ct	G	OP	40 29	fddiPORT 41
fddiPORTLCTFail-Ct	G	MP	40 2A	fddiPORT 42
fddiPORTLerGrp	**G**	**M**	**40 32**	**fddiPORT 50**
fddiPORTLer-Estimate	G	MP	40 33	fddiPORT 51
fddiPORTLem-Reject-Ct	G	MP	40 34	fddiPORT 52
fddiPORTLem-Ct	G	MP	40 35	fddiPORT 53
fddiPORTLer-Cutoff	GR	M	40 3A	fddiPORT 58
fddiPORTLer-Alarm	GR	M	40 3B	fddiPORT 59
fddiPORTStatusGrp	**G**	**M**	**40 3C**	**fddiPORT 60**
fddiPORTConnectState	G	MP	40 3D	fddiPORT 61
fddiPORTPCMState	G	MP	40 3E	fddiPORT 62
fddiPORTPC-Withhold	G	MP	40 3F	fddiPORT 63
fddiPORTLerFlag	G	MP	40 40	fddiPORT 64
fddiPORTHardwarePresent	G	M	40 41	fddiPORT 65
Not Grouped				
fddiPORTVendorAttrib	GR	O	40 FF	fddiPORT 255

6.4.5.1.5 Actions

SMT Actions				
fddiSMTStationAction	S	M	10 3C	fddiSMT 60
fddiSMTVendorAction	S	O	10 FE	fddiSMT 254
MAC Actions				
fddiMACVendorAction	S	O	20 FE	fddiMAC 254
PATH Actions				
fddiPATHVendorAction	S	O	32 FE	fddiPATH 254
PORT Actions				
fddiPORTAction	S	M	40 46	fddiPORT 70
fddiPORTVendorAction	S	O	40 FE	fddiPORT 254

6.4.5.1.6 Notifications

SMT Notifications				
fddiSMTHoldCondition	NA	O	10 47	fddiSMT 71
fddiSMTPeerWrapCondition	NA	M	10 48	fddiSMT 72
fddiSMTVendorNotification	NA	O	10 FC	fddiSMT 252
MAC Notifications				
fddiMACDuplicateAddressCondition	NA	M	20 8C	fddiMAC 140
fddiMACFrameErrorCondition	NA	M	20 8D	fddiMAC 141
fddiMACNotCopiedCondition	NA	O	20 8E	fddiMAC 142
fddiMACNeighbourChangeEvent	NA	M	20 8F	fddiMAC 143
fddiMACPathChangeEvent	NA	M	20 90	fddiMAC 144
fddiMACVendorNotification	NA	O	20 FC	fddiMAC 252
PATH Notifications				
fddiPATHVendorNotification	NA	O	32 FC	fddiPATH 252

PORT Notifications				
fddiPORTLerCondition	NA	M	40 50	fddiPORT 80
fddiPORTUndesiredConnectionAttemptEvent	NA	M	40 51	fddiPORT 81
fddiPORTEBErrorCondition	NA	O	40 52	fddiPORT 82
fddiPORTPathChangeEvent	NA	M	40 53	fddiPORT 83
fddiPORTVendorNotification	NA	O	40 FC	fddiPORT 252

6.4.5.2 Managed object class templates

6.4.5.2.1 SMT object class

fddiSMT **MANAGED OBJECT CLASS**

DERIVED FROM "Rec.X.71|ISO/IEC 10165-2":top;
CHARACTERIZED BY

fddiSMTBase **PACKAGE**

BEHAVIOUR

fddiSMTBaseBhv BEHAVIOUR

DEFINED AS "The fddiSMT object class provides the support necessary at the station (node) level to manage the processes underway in the various FDDI layers such that a station (node) may work cooperatively as a part of an FDDI network. The fddiSMT object class provides services such as connection management, station insertion and removal, station initialization, configuration management, fault isolation and recovery, communication protocol for external authority, scheduling policies, and collection of statistics.";;

ATTRIBUTES

fddiSMTStationId	GET,
fddiSMTOpVersionId	GET,
fddiSMTHiVersionId	GET,
fddiSMTLoVersionId	GET,
fddiSMTUserData	GET-REPLACE,
fddiSMTMIBVersionId	GET,
fddiSMTMAC-Ct	GET,
fddiSMTNonMaster-Ct	GET,
fddiSMTMaster-Ct	GET,
fddiSMTAvailablePaths	GET,
fddiSMTConfigCapabilities	GET,
fddiSMTConfigPolicy	GET-REPLACE,
fddiSMTConnectionPolicy	GET-REPLACE,
fddiSMTT-Notify	GET-REPLACE,
fddiSMTStatRptPolicy	GET-REPLACE,
fddiSMTTrace-MaxExpiration	GET-REPLACE,
fddiSMTPORTIndexes	GET,
fddiSMTMACIndexes	GET,
fddiSMTBypassPresent	GET,
fddiSMTECMState	GET,
fddiSMTCF-State	GET,
fddiSMTRemoteDisconnectFlag	GET,
fddiSMTStationStatus	GET,
fddiSMTPeerWrapFlag	GET,
fddiSMTTimeStamp	GET,
fddiSMTTransitionTimeStamp	GET;

ATTRIBUTE GROUPS

fddiSMTStationIdGrp,
fddiSMTStationConfigGrp,
fddiSMTStatusGrp,
fddiSMTMIBOperationGrp;

ACTIONS

fddiSMTStationAction;

NOTIFICATIONS

fddiSMTPeerWrapCondition;;;

```
CONDITIONAL PACKAGES

    manufacturerdata                        PACKAGE
        BEHAVIOUR
            manufacturerdataBhv BEHAVIOUR
                DEFINED AS "This package provides support for the Manufacturer data
                attribute of a node.";;
            ATTRIBUTES
                fddiSMTManufacturerData     GET;
            ATTRIBUTE GROUPS
                fddiSMTStationIdGrp
                    fddiSMTManufacturerData;
            REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMTpkg(1)
                manufacturerdata(1)};
        PRESENT IF                          "Manufacturer Data Supported",

    parametermanagement                     PACKAGE
        BEHAVIOUR
            parametermanagementBhv BEHAVIOUR
                DEFINED AS "This package provides support for the parameter management
                capability of a node.";;
            ATTRIBUTES
                fddiSMTSetCount             GET,
                fddiSMTLastSetStationId     GET;
            ATTRIBUTE GROUPS
                fddiSMTMIBOperationGrp
                    fddiSMTSetCount
                    fddiSMTLastSetStationId;
            REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMTpkg(1)
                parametermanagement(2)};
        PRESENT IF                          "Parameter Management Frame Set
                                            operations are supported",

    hold                                    PACKAGE
        BEHAVIOUR
            holdBhv BEHAVIOUR
                DEFINED AS "This package provides support for the Hold policy. The Hold
                policy only applies to dual attach, dual MAC nodes.";;
            ATTRIBUTES
                fddiSMTHoldState            GET;
            ATTRIBUTE GROUPS
                fddiSMTStatusGrp
                    fddiSMTHoldState;
            NOTIFICATIONS
                fddiSMTHoldCondition;
            REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMTpkg(1)
                hold(3)};
        PRESENT IF                          "Hold policy supported",

    smtvendorspecific                       PACKAGE
        BEHAVIOUR
            smtvendorspecificBhv BEHAVIOUR
                DEFINED AS "This package provides support for the extension of the MIB for
                the use of specific vendors.";;
            ATTRIBUTES
                fddiSMTVendorAttrib         GET-REPLACE;
            ACTIONS
                fddiSMTVendorAction;
            NOTIFICATIONS
                fddiSMTVendorNotification;
```

```
            REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMTpkg(1)
                smtvendorspecific(4)};
        PRESENT IF                                  "Vendor Extensions Supported";
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)};
```

6.4.5.2.2 MAC object class

```
fddiMAC                                     MANAGED OBJECT CLASS
    DERIVED FROM "Rec.X.71|ISO/IEC 10165-2":top;
    CHARACTERIZED BY

        fddiMACBase                         PACKAGE
            BEHAVIOUR
                fddiMACBaseBhv BEHAVIOUR
                    DEFINED AS "The fddiMAC object provides support for the access control of
                    the medium and generation and verification of frame check sequences to assure
                    the proper delivery of valid data to the higher layers. It also concerns itself with
                    the generation and recognition of device addresses and the peer-to-peer
                    associations within the FDDI network.";;
                ATTRIBUTES
                    fddiMACIndex                    GET,
                    fddiMACRequestedPaths           GET-REPLACE,
                    fddiMACFrameErrorThreshold      GET-REPLACE,
                    fddiMACHardwarePresent          GET,
                    fddiMACMA-UnitdataEnable        GET-REPLACE;
                ATTRIBUTE GROUPS
                    fddiMACConfigGrp,
                    fddiMACFrameErrorConditionGrp,
                    fddiMACStatusGrp;
                NOTIFICATIONS
                    fddiMACDuplicateAddressCondition,
                    fddiMACFrameErrorCondition,
                    fddiMACNeighbourChangeEvent,
                    fddiMACPathChangeEvent;;;

    CONDITIONAL PACKAGES

        machardwarepresent                  PACKAGE
            BEHAVIOUR
                machardwarepresentBhv BEHAVIOUR
                    DEFINED AS "This package contains attributes which support the management
                    of the MAC entity within a station. This package shall be included in every MAC
                    object where the underlying hardware for the MAC entity is present.";;
                ATTRIBUTES
                    fddiMACFrameStatusFunctions     GET,
                    fddiMACT-MaxCapability          GET,
                    fddiMACTVXCapability            GET,
                    fddiMACAvailablePaths           GET,
                    fddiMACCurrentPath              GET,
                    fddiMACUpstreamNbr              GET,
                    fddiMACDownstreamNbr            GET,
                    fddiMACOldUpstreamNbr           GET,
                    fddiMACOldDownstreamNbr         GET,
                    fddiMACDupAddressTest           GET,
                    fddiMACDownstreamPORTType       GET,
                    fddiMACSMTAddress               GET,
                    fddiMACT-Req                    GET,
                    fddiMACT-Neg                    GET,
                    fddiMACT-Max                    GET,
                    fddiMACTvxValue                 GET,
```

```
            fddiMACFrame-Ct                 GET,
            fddiMACCopied-Ct                GET,
            fddiMACTransmit-Ct              GET,
            fddiMACError-Ct                 GET,
            fddiMACLost-Ct                  GET,
            fddiMACFrameErrorRatio          GET,
            fddiMACRMTState                 GET,
            fddiMACDA-Flag                  GET,
            fddiMACUNDA-Flag                GET,
            fddiMACFrameErrorFlag           GET,
            fddiMACMA-UnitdataAvailable     GET;

        ATTRIBUTE GROUPS
            fddiMACCapabilitiesGrp
                fddiMACFrameStatusFunctions
                fddiMACT-MaxCapability
                fddiMACTVXCapability,
            fddiMACConfigGrp
                fddiMACAvailablePaths
                fddiMACCurrentPath
                fddiMACUpstreamNbr
                fddiMACDownstreamNbr
                fddiMACOldUpstreamNbr
                fddiMACOldDownstreamNbr
                fddiMACDupAddressTest
                fddiMACDownstreamPORTType,
            fddiMACAddressGrp
                fddiMACSMTAddress,
            fddiMACOperationGrp
                fddiMACT-Req
                fddiMACT-Neg
                fddiMACT-Max
                fddiMACTvxValue,
            fddiMACCountersGrp
                fddiMACFrame-Ct
                fddiMACCopied-Ct
                fddiMACTransmit-Ct
                fddiMACError-Ct
                fddiMACLost-Ct,
            fddiMACFrameErrorConditionGrp
                fddiMACFrameErrorRatio,
            fddiMACStatusGrp
                fddiMACRMTState
                fddiMACDA-Flag
                fddiMACUNDA-Flag
                fddiMACFrameErrorFlag
                fddiMACMA-UnitdataAvailable;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            machardwarepresent(1)};
    PRESENT IF                          "Underlying hardware support of the MAC
                                        object is present.",

    framenotcopied                      PACKAGE
        BEHAVIOUR
            framenotcopiedBhv BEHAVIOUR
                DEFINED AS "This package provides support for the detection and reporting of
                the Frame Not Copied condition of the fddiMAC object.";;
```

```
        ATTRIBUTES
            fddiMACNotCopied-Ct             GET,
            fddiMACNotCopiedRatio           GET,
            fddiMACNotCopiedFlag            GET;
        ATTRIBUTE GROUPS
            fddiMACNotCopiedConditionGrp
                fddiMACNotCopiedRatio,
            fddiMACCountersGrp
                fddiMACNotCopied-Ct,
            fddiMACStatusGrp
                fddiMACNotCopiedFlag;
        NOTIFICATIONS
            fddiMACNotCopiedCondition;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            framenotcopied(2)};
PRESENT IF                                  "Frame not copied is supported AND
                                            underlying hardware support is present.",

framenotcopiedmanagement                    PACKAGE
    BEHAVIOUR
        framenotcopiedmanagementBhv BEHAVIOUR
            DEFINED AS "This package provides support for the detection and reporting of
            the Frame Not Copied condition of the fddiMAC object.";;
        ATTRIBUTES
            fddiMACNotCopiedThreshold       GET-REPLACE;
        ATTRIBUTE GROUPS
            fddiMACNotCopiedConditionGrp
                fddiMACNotCopiedThreshold;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            framenotcopiedmanagement(3)};
PRESENT IF                                  "Frame not copied is supported.",

prioritythreshold                           PACKAGE
    BEHAVIOUR
        prioritythresholdBhv BEHAVIOUR
            DEFINED AS "This package provides support for the priority thresholding
            capability of the fddiMAC object.";;
        ATTRIBUTES
            fddiMACT-Pri0                   GET,
            fddiMACT-Pri1                   GET,
            fddiMACT-Pri2                   GET,
            fddiMACT-Pri3                   GET,
            fddiMACT-Pri4                   GET,
            fddiMACT-Pri5                   GET,
            fddiMACT-Pri6                   GET;
        ATTRIBUTE GROUPS
            fddiMACOperationGrp
                fddiMACT-Pri0
                fddiMACT-Pri1
                fddiMACT-Pri2
                fddiMACT-Pri3
                fddiMACT-Pri4
                fddiMACT-Pri5
                fddiMACT-Pri6;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            prioritythreshold(4)};
PRESENT IF                                  "Priority Thresholding supported AND
                                            underlying hardware support is present",
```

```
tvxexpired                                  PACKAGE
    BEHAVIOUR
        tvxexpiredBhv BEHAVIOUR
            DEFINED AS "This package provides support for monitoring the expiration of
            TVX.";;
        ATTRIBUTES
            fddiMACTvxExpired-Ct            GET;
        ATTRIBUTE GROUPS
            fddiMACCountersGrp
                fddiMACTvxExpired-Ct;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            tvxexpired(5)};
PRESENT IF                                  "TVX Expiration monitored AND underlying
                                            hardware support is present",

latecount                                   PACKAGE
    BEHAVIOUR
        latecountBhv BEHAVIOUR
            DEFINED AS "This package provides support for monitoring the number of TRT
            expirations since MAC was reset or a token was received.";;
        ATTRIBUTES
            fddiMACLate-Ct                  GET;
        ATTRIBUTE GROUPS
            fddiMACCountersGrp
                fddiMACLate-Ct;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            latecount(6)};
PRESENT IF                                  "Late Counter monitored AND underlying
                                            hardware support is present",

ringopcount                                 PACKAGE
    BEHAVIOUR
        ringopcountBhv BEHAVIOUR
            DEFINED AS "This package provides support for monitoring the number of
            times the ring has entered the 'Ring_Operational' state from the 'Ring Not
            Operational' state.";;
        ATTRIBUTES
            fddiMACRingOp-Ct                GET;
        ATTRIBUTE GROUPS
            fddiMACCountersGrp
                fddiMACRingOp-Ct;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            ringopcount(7)};
PRESENT IF                                  "RingOp Counter monitored AND underlying
                                            hardware support is present",

longgrpaddress                              PACKAGE
    BEHAVIOUR
        longgrpaddressBhv BEHAVIOUR
            DEFINED AS "This package provides support for 48 bit group addresses for the
            MAC.";;
        ATTRIBUTES
            fddiMACLongGrpAddress           GET;
        ATTRIBUTE GROUPS
            fddiMACAddressGrp
                fddiMACLongGrpAddress;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            longgrpaddress(8)};
PRESENT IF                                  "Long Group Addresses supported AND
                                            underlying hardware support is present.",
```

```
shortgrpaddress                          PACKAGE
    BEHAVIOUR
        shortgrpaddressBhv BEHAVIOUR
            DEFINED AS "This package provides support for 16 bit group addresses for the
            MAC.";;
        ATTRIBUTES
            fddiMACShortGrpAddress       GET;
        ATTRIBUTE GROUPS
            fddiMACAddressGrp
                fddiMACShortGrpAddress;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            shortgrpaddress(9)};
    PRESENT IF                           "Short Group Addresses supported AND
                                         underlying hardware support is present.",

tokencounter                             PACKAGE
    BEHAVIOUR
        tokencounterBhv BEHAVIOUR
            DEFINED AS "This package provides support for counting the tokens received
            by a  station.";;
        ATTRIBUTES
            fddiMACToken-Ct              GET;
        ATTRIBUTE GROUPS
            fddiMACCountersGrp
                fddiMACToken-Ct;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            tokencounter(10)};
    PRESENT IF                           "Token Counting supported AND underlying
                                         hardware support is present",

bridgefunctions                          PACKAGE
    BEHAVIOUR
        bridgefunctionsBhv BEHAVIOUR
            DEFINED AS "This package provides support for a MAC performing bridge
            functions.";;
        ATTRIBUTES
            fddiMACBridgeFunctions       GET;
        ATTRIBUTE GROUPS
            fddiMACCapabilitiesGrp
                fddiMACBridgeFunctions;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            bridgefunctions(11)};
    PRESENT IF                           "Bridge Functions supported AND underlying
                                         hardware support is present",

macvendorspecific                        PACKAGE
    BEHAVIOUR
        macvendorspecificBhv BEHAVIOUR
            DEFINED AS "This package provides support for the extension of the MIB for
            the use of specific vendors.";;
        ATTRIBUTES
            fddiMACVendorAttrib          GET-REPLACE;
        ACTIONS
            fddiMACVendorAction;
        NOTIFICATIONS
            fddiMACVendorNotification;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMACpkg(2)
            macvendorspecific(12)};
    PRESENT IF                           "Vendor Extensions supported";;
;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32)};
```

6.4.5.2.3 PATH object class

```
fddiPATH                                        MANAGED OBJECT CLASS
    DERIVED FROM "Rec.X.71|ISO/IEC 10165-2":top;
    CHARACTERIZED BY

        fddiPATHBase                            PACKAGE
            BEHAVIOUR
                fddiPATHBaseBhv BEHAVIOUR
                    DEFINED AS "The fddiPATH object provides for the management of the Path
                    entity. A station may have multiple Paths. Each instance of this class models the
                    configuration of a station's internal resources (e.g. MACs and PORTs) as they
                    relate to an external ring (see 9.7.2).";;
                ATTRIBUTES
                    fddiPATHIndex               GET,
                    fddiPATHConfiguration       GET,
                    fddiPATHTVXLowerBound       GET-REPLACE,
                    fddiPATHT-MaxLowerBound     GET-REPLACE,
                    fddiPATHMaxT-Req            GET-REPLACE;
                ATTRIBUTE GROUPS
                    fddiPATHConfigGrp;;;

    CONDITIONAL PACKAGES

        synchbandwidthrequestor                 PACKAGE
            BEHAVIOUR
                synchbandwidthrequestorBhv BEHAVIOUR
                    DEFINED AS "This package provides support for the requisition of synchronous
                    bandwidth.";;
                ATTRIBUTES
                    fddiPATHSbaPayload          GET-REPLACE,
                    fddiPATHSbaOverhead         GET-REPLACE;
                ATTRIBUTE GROUPS
                    fddiPATHConfigGrp
                        fddiPATHSbaPayload
                        fddiPATHSbaOverhead;
                REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATHpkg(3)
                    synchbandwidthrequestor(1)};
            PRESENT IF                          "Synchronous service supported",

        synchbandwidthmgt                       PACKAGE
            BEHAVIOUR
                synchbandwidthmgtBhv BEHAVIOUR
                    DEFINED AS "This package provides support for the allocation of synchronous
                    bandwidth.";;
                ATTRIBUTES
                    fddiPATHSbaAvailable        GET-REPLACE;
                ATTRIBUTE GROUPS
                    fddiPATHConfigGrp
                        fddiPATHSbaAvailable;
                REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATHpkg(3)
                    synchbandwidthmgt(2)};
            PRESENT IF                          "Allocating Synchronous bandwidth",

        ringlatency                             PACKAGE
            BEHAVIOUR
                ringlatencyBhv BEHAVIOUR
                    DEFINED AS "This package provides support for measuring the total
                    accumulated latency of the ring associated with this path.";;
                ATTRIBUTES
                    fddiPATHRingLatency         GET-REPLACE;
```

```
        ATTRIBUTE GROUPS
            fddiPATHConfigGrp
                fddiPATHRingLatency;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATHpkg(3)
            ringlatency(3)};
    PRESENT IF                              "Latency measurement supported",

    tracestatus                             PACKAGE
        BEHAVIOUR
            tracestatusBhv BEHAVIOUR
                DEFINED AS "This package provides status from the trace function.";;
            ATTRIBUTES
                fddiPATHTraceStatus         GET;
            ATTRIBUTE GROUPS
                fddiPATHConfigGrp
                    fddiPATHTraceStatus;
            REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATHpkg(3)
                tracestatus(4)};
    PRESENT IF                              "Trace Status is supported",

    restrictedtoken                         PACKAGE
        BEHAVIOUR
            restrictedtokenBhv BEHAVIOUR
                DEFINED AS "This package along with the RMT optional protocol provides
                support for monitoring the restricted token service.";;
            ATTRIBUTES
                fddiPATHT-Rmode             GET-REPLACE;
            ATTRIBUTE GROUPS
                fddiPATHConfigGrp
                    fddiPATHT-Rmode;
            REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATHpkg(3)
                restrictedtoken(5)};
    PRESENT IF                              "Restricted Token Monitoring is supported",

    pathvendorspecific                      PACKAGE
        BEHAVIOUR
            pathvendorspecificBhv BEHAVIOUR
                DEFINED AS "This package provides support for the extension of the MIB for
                the use of specific vendors.";;
            ATTRIBUTES
                fddiPATHVendorAttrib        GET-REPLACE;
            ACTIONS
                fddiPATHVendorAction;
            NOTIFICATIONS
                fddiPATHVendorNotification;
            REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATHpkg(3)
                pathvendorspecific(6)};
        PRESENT IF                          "Vendor Extensions Supported";;
        ;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50)};
```

6.4.5.2.4 PORT object class

```
fddiPORT                                    MANAGED OBJECT CLASS
    DERIVED FROM "Rec.X.71|ISO/IEC 10165-2":top;
    CHARACTERIZED BY
```

fddiPORTBase **PACKAGE**
BEHAVIOUR
fddiPORTBaseBhv BEHAVIOUR
DEFINED AS "The fddiPORT object provides support for the medium, connector, bypassing, and driver/receiver requirements. It also defines encode/decode and clock requirements as required for framing the data for transmission on the medium by the higher layers of the FDDI.";;
ATTRIBUTES
fddiPORTMy-Type GET,
fddiPORTConnectionPolicies GET-REPLACE,
fddiPORTRequestedPaths GET-REPLACE,
fddiPORTIndex GET,
fddiPORTLer-Cutoff GET-REPLACE,
fddiPORTLer-Alarm GET-REPLACE,
fddiPORTHardwarePresent GET;
ATTRIBUTE GROUPS
fddiPORTConfigGrp,
fddiPORTLerGrp,
fddiPORTStatusGrp;
ACTIONS
fddiPORTAction;
NOTIFICATIONS
fddiPORTLerCondition,
fddiPORTUndesiredConnectionAttemptEvent,
fddiPORTPathChangeEvent;;;

CONDITIONAL PACKAGES

porthardwarepresent **PACKAGE**
BEHAVIOUR
porthardwarepresentBhv BEHAVIOUR
DEFINED AS "This package contains attributes which support the management of the PORT entity within a station. This package shall be included in every PORT object where the underlying hardware for the PORT entity is present.";;
ATTRIBUTES
fddiPORTNeighbourType GET,
fddiPORTMACIndicated GET,
fddiPORTCurrentPath GET,
fddiPORTMACPlacement GET,
fddiPORTAvailablePaths GET,
fddiPORTPMDClass GET,
fddiPORTConnectionCapabilities GET,
fddiPORTBS-Flag GET,
fddiPORTLCTFail-Ct GET,
fddiPORTLer-Estimate GET,
fddiPORTLem-Reject-Ct GET,
fddiPORTLem-Ct GET,
fddiPORTConnectState GET,
fddiPORTPCMState GET,
fddiPORTPC-Withhold GET,
fddiPORTLerFlag GET;
ATTRIBUTE GROUPS
fddiPORTConfigGrp
fddiPORTNeighbourType
fddiPORTMACIndicated
fddiPORTCurrentPath
fddiPORTMACPlacement
fddiPORTAvailablePaths
fddiPORTPMDClass
fddiPORTConnectionCapabilities,

```
            fddiPORTOperationGrp
                fddiPORTBS-Flag,
            fddiPORTErrorCtrsGrp
                fddiPORTLCTFail-Ct,
            fddiPORTLerGrp
                fddiPORTLer-Estimate
                fddiPORTLem-Reject-Ct
                fddiPORTLem-Ct,
            fddiPORTStatusGrp
                fddiPORTConnectState
                fddiPORTPCMState
                fddiPORTPC-Withhold
                fddiPORTLerFlag;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORTpkg(4)
            porthardwarepresent(1)};
PRESENT IF                              "Underlying hardware support of the port
                                        object is present.",

eberrorcounter                          PACKAGE
    BEHAVIOUR
        eberrorcounterBhv BEHAVIOUR
            DEFINED AS "This package provides support for counting elasticity buffer errors
            in the fddiPORT object.";;
        ATTRIBUTES
            fddiPORTEBError-Ct          GET;
        ATTRIBUTE GROUPS
            fddiPORTErrorCtrsGrp
                fddiPORTEBError-Ct;
        NOTIFICATIONS
            fddiPORTEBErrorCondition;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORTpkg(4)
            eberrorcounter(2)};
PRESENT IF                              "EB Error Tracking supported AND
                                        underlying hardware support is present.",

maclocalloop                            PACKAGE
    BEHAVIOUR
        maclocalloopBhv BEHAVIOUR
            DEFINED AS "This package provides support for the time for the optional MAC
            Local Loop.";;
        ATTRIBUTES
            fddiPORTMACLoop-Time        GET-REPLACE;
        ATTRIBUTE GROUPS
            fddiPORTConfigGrp
                fddiPORTMACLoop-Time;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORTpkg(4)
            maclocalloop(3)};
PRESENT IF                              "MAC Local Loop supported",

maint-LS                                PACKAGE
    BEHAVIOUR
        maint-LSBhv BEHAVIOUR
            DEFINED AS "This package provides support for identifying the state of the
            line.";;
        ATTRIBUTES
            fddiPORTPC-LS               GET;
        ATTRIBUTE GROUPS
            fddiPORTOperationGrp
                fddiPORTPC-LS;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORTpkg(4)
            maint-LS(4)};
```

```
        PRESENT IF                              "Maint_LS is supported AND underlying
                                                hardware support is present.",

        maint-LSmanagement                      PACKAGE
            BEHAVIOUR
                maint-LSmanagementBhv BEHAVIOUR
                    DEFINED AS "This package provides support for identifying the state of the
                    line.";;
                ATTRIBUTES
                    fddiPORTMaint-LS            GET-REPLACE;
                ATTRIBUTE GROUPS
                    fddiPORTOperationGrp
                        fddiPORTMaint-LS;
                REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORTpkg(4)
                    maint-LSmanagement(5)};
        PRESENT IF                              "Maint_LS is supported.",

        portvendorspecific                      PACKAGE
            BEHAVIOUR
                portvendorspecific Bhv BEHAVIOUR
                    DEFINED AS "This package provides support for the extension of the MIB for
                    the use of specific vendors.";;
                ATTRIBUTES
                    fddiPORTVendorAttrib        GET-REPLACE;
                ACTIONS
                    fddiPORTVendorAction;
                NOTIFICATIONS
                    fddiPORTVendorNotification;
                REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORTpkg(4)
                    portvendorspecific(6)};
        PRESENT IF                              "Vendor Extensions Supported";;
    ;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64)};
```

6.4.5.3 Attribute group templates

6.4.5.3.1 SMT attribute groups

```
fddiSMTStationIdGrp                         ATTRIBUTE GROUP
        GROUP ELEMENTS
            fddiSMTStationId,
            fddiSMTOpVersionId,
            fddiSMTHiVersionId,
            fddiSMTLoVersionId,
            fddiSMTUserData,
            fddiSMTMIBVersionId;
        DESCRIPTION
            The identification group includes the attributes that identify the station for its operation
            in FDDI.;
        REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)
            fddiSMTStationIdGrp(10)};

fddiSMTStationConfigGrp                     ATTRIBUTE GROUP
        GROUP ELEMENTS
            fddiSMTMAC-Ct,
            fddiSMTNonMaster-Ct,
            fddiSMTMaster-Ct,
            fddiSMTAvailablePaths,
            fddiSMTConfigCapabilities,
            fddiSMTConfigPolicy,
            fddiSMTConnectionPolicy,
```

fddiSMTT-Notify,
fddiSMTStatRptPolicy,
fddiSMTTrace-MaxExpiration,
fddiSMTPORTIndexes,
fddiSMTMACIndexes,
fddiSMTBypassPresent;
DESCRIPTION
The configuration group includes the attributes that characterize the station's configuration.;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTStationConfigGrp(20)};

fddiSMTStatusGrp **ATTRIBUTE GROUP**
GROUP ELEMENTS
fddiSMTECMState,
fddiSMTCF-State,
fddiSMTRemoteDisconnectFlag,
fddiSMTStationStatus,
fddiSMTPeerWrapFlag;
DESCRIPTION
The status group includes attributes that define the general station state.;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTStatusGrp(40)};

fddiSMTMIBOperationGrp **ATTRIBUTE GROUP**
GROUP ELEMENTS
fddiSMTTimeStamp,
fddiSMTTransitionTimeStamp;
DESCRIPTION
The MIB operation group includes the attributes that are associated with MIB accesses. The last three attributes in the group, the TransitionTimeStamp, SetCount, and LastSetStationId are used in the Parameter Management Frame (PMF) protocol (see 8.4).;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTMIBOperationGrp(50)};

6.4.5.3.2 MAC attribute groups

fddiMACCapabilitiesGrp **ATTRIBUTE GROUP**
DESCRIPTION
The MAC capabilities group includes attributes that describe the frame status handling, bridge, and timer bounds capabilities.;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACCapabilitiesGrp(10)};

fddiMACConfigGrp **ATTRIBUTE GROUP**
GROUP ELEMENTS
fddiMACIndex;
DESCRIPTION
The MAC configuration group includes the Path configuration attributes as well as neighbour attributes that give a topological characterization of the MAC in the station and in the network.;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACConfigGrp(20)};

fddiMACAddressGrp **ATTRIBUTE GROUP**
DESCRIPTION
The address group includes all of the local address attributes for the MAC.;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACAddressGrp(40)};

fddiMACOperationGrp **ATTRIBUTE GROUP**

DESCRIPTION

The operation group includes all of the MAC operational timer attributes and the operational FrameStatus setting functionality.;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACOperationGrp(50)};

fddiMACCountersGrp **ATTRIBUTE GROUP**

DESCRIPTION

The MAC counters group includes counts of frames recognized by, transmitted and received by the MAC, and counters that are valuable for fault detection and performance evaluation.;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACCountersGrp(70)};

fddiMACFrameErrorConditionGrp **ATTRIBUTE GROUP**

DESCRIPTION

The MAC frame error condition group includes control and status attributes of the frame error condition reported in the Status Reporting Protocol (see 7.2.7 and 8.3).;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACFrameErrorConditionGrp(90)};

fddiMACNotCopiedConditionGrp **ATTRIBUTE GROUP**

DESCRIPTION

The not copied condition group includes control and status attributes of the not copied condition reported in the Status Reporting Protocol (see 7.2.7 and 8.3).;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACNotCopiedConditionGrp(100)};

fddiMACStatusGrp **ATTRIBUTE GROUP**

GROUP ELEMENTS

fddiMACHardwarePresent,
fddiMACMA-UnitdataEnable;

DESCRIPTION

The status group includes general state and condition attributes for the MAC.;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACStatusGrp(110)};

6.4.5.3.3 PATH attribute groups

fddiPATHConfigGrp **ATTRIBUTE GROUP**

GROUP ELEMENTS

fddiPATHIndex,
fddiPATHConfiguration,
fddiPATHTVXLowerBound,
fddiPATHT-MaxLowerBound,
fddiPATHMaxT-Req;

DESCRIPTION

The Path configuration group includes attributes that define the configuration of a Path, and control the operation of objects inserted on the Path.;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHConfigGrp(10)};

6.4.5.3.4 PORT attribute groups

fddiPORTConfigGrp **ATTRIBUTE GROUP**

GROUP ELEMENTS

fddiPORTIndex,
fddiPORTMy-Type;

DESCRIPTION

The Port configuration group includes status and control attributes that describe the configuration of a Port.;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64)
fddiPORTConfigGrp(10)};

fddiPORTOperationGrp **ATTRIBUTE GROUP**
DESCRIPTION
The Port operation group includes attributes that affect the operation of the Port.;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64)
fddiPORTOperationGrp(30)};

fddiPORTErrorCtrsGrp **ATTRIBUTE GROUP**
DESCRIPTION
The error counter group includes Port error counters that are not
specifically related to the Link Error Monitoring Function.;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64)
fddiPORTErrorCtrsGrp(40)};

fddiPORTLerGrp **ATTRIBUTE GROUP**
DESCRIPTION
The link error reporting status group includes the status and control attributes used to
determine when a link error threshold has been exceeded (see 9.4.7).;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTLerGrp(50)};

fddiPORTStatusGrp **ATTRIBUTE GROUP**
GROUP ELEMENTS
fddiPORTHardwarePresent;
DESCRIPTION
The port status group includes general state attributes for the Port;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64)
fddiPORTStatusGrp(60)};

6.4.5.4 Attribute templates

6.4.5.4.1 SMT attributes

fddiSMTStationId **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.StationIdType;
BEHAVIOUR
fddiSMTStationIdBhv BEHAVIOUR
DEFINED AS Used to uniquely identify an FDDI station.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTStationId(11)};

fddiSMTOpVersionId **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.OpVersionIdType;
BEHAVIOUR
fddiSMTOpVersionIdBhv BEHAVIOUR
DEFINED AS "The version that this station is using for its operation (see 7.1.2.2). The
value of this attribute is 2 for this SMT revision.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)
fddiSMTOpVersionId(13)};

fddiSMTHiVersionId **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.HiVersionIdType;
BEHAVIOUR
fddiSMTHiVersionIdBhv BEHAVIOUR
DEFINED AS "The highest version of SMT that this station supports (see 7.1.2.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTHiVersionId(14)};

fddiSMTLoVersionId **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.LoVersionIdType;
BEHAVIOUR
fddiSMTLoVersionIdBhv BEHAVIOUR
DEFINED AS "The lowest version of SMT that this station supports (see 7.1.2.2).";;

```
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)
                    fddiSMTLoVersionId(15)};

fddiSMTManufacturerData                         ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-SMT.ManufacturerDataType;
    BEHAVIOUR
        fddiSMTManufacturerDataBhv BEHAVIOUR
            DEFINED AS    "32 octets of manufacturer data. The first component, manufacturerOUI, is
            the three octet Organizationally Unique Identifier (OUI) assigned by IEEE. The second
            component, manufacturerData, is the manufacturer's data (29 octets).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)
                    fddiSMTManufacturerData(16)};

fddiSMTUserData                                 ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-SMT.UserDataType;
    BEHAVIOUR
        fddiSMTUserDataBhv BEHAVIOUR
            DEFINED AS    "This attribute contains 32 octets of user defined information. The
            information shall be an ASCII string.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTUserData(17)};

fddiSMTMIBVersionId                             ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-SMT.MIBVersionIdType;
    BEHAVIOUR
        fddiSMTMIBVersionIdBhv BEHAVIOUR
            DEFINED AS    "The version of the FDDI MIB of this station.  The value of this attribute is
            1 for this SMT revision.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)
                    fddiSMTMIBVersionId(18)};

fddiSMTMac-Ct                                   ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-SMT.Mac-CtType;
    BEHAVIOUR
        fddiSMTMac-CtBhv BEHAVIOUR
            DEFINED AS    "The number of MACs in this station or concentrator.  This number
            includes MAC instances that are not supported by underlying hardware.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTMac-Ct(21)};

fddiSMTNonMaster-Ct                             ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-SMT.NonMaster-CtType;
    BEHAVIOUR
        fddiSMTNonMaster-CtBhv BEHAVIOUR
            DEFINED AS    "The value of this attribute is the number of A, B, and S ports in this
            station or concentrator.  This number includes PORT instances that are not supported by
            underlying hardware.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTNonMaster-
                    Ct(22)};

fddiSMTMaster-Ct                                ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-SMT.Master-CtType;
    BEHAVIOUR
        fddiSMTMaster-CtBhv BEHAVIOUR
            DEFINED AS    "The number of M Ports in a node. If the node is not a concentrator, the
            value of the attribute is zero.  This number includes PORT instances that are not
            supported by underlying hardware.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTMaster-Ct(23)};

fddiSMTAvailablePaths                           ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-SMT.AvailablePathsType;
    BEHAVIOUR
        fddiSMTAvailablePathsBhv BEHAVIOUR
            DEFINED AS    "A bit string that indicates the path types that are available in the station.";;
```

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTAvailablePaths(24)};

fddiSMTConfigCapabilities **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.ConfigCapabilitiesType;
BEHAVIOUR
fddiSMTConfigCapabilitiesBhv BEHAVIOUR
DEFINED AS "A bit string that indicates the configuration capabilities of a node. The 'Hold Available' bit indicates the support of the optional Hold Function, which is controlled by fddiSMTConfigPolicy. The 'CF-Wrap-AB' bit indicates that the station has the capability of performing a wrap_ab (see 9.7.2.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTConfigCapabilities(25)};

fddiSMTConfigPolicy **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.ConfigPolicyType;
BEHAVIOUR
fddiSMTConfigPolicyBhv BEHAVIOUR
DEFINED AS "A bit string representing the configuration policies currently desired in a node. 'Hold' is one of the terms used for the Hold Flag (see 9.4.3.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTConfigPolicy(26)};

fddiSMTConnectionPolicy **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.ConnectionPolicyType;
BEHAVIOUR
fddiSMTConnectionPolicyBhv BEHAVIOUR
DEFINED AS "A bit string representing the connection policies in effect in a node. A station sets the corresponding bit for each of the connection types that it rejects. The letter designations, X and Y, in the 'rejectX-Y' names have the following significance: X represents the PC-Type of the local PORT and Y represents the PC_Type of the adjacent PORT (PC_Neighbour). The evaluation of Connection-Policy (PC-Type, PC-Neighbour) is done to determine the setting of T-Val(3) in the PC-Signalling sequence (see 9.6.3). Note that Bit 15, (rejectM-M), is always set and cannot be cleared.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTConnectionPolicy(27)};

fddiSMTT-Notify **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.T-NotifyType;
BEHAVIOUR
fddiSMTT-NotifyBhv BEHAVIOUR
DEFINED AS "The timer, expressed in seconds, used in the Neighbour Notification protocol. It has a range of 2 s to 30 s, and its default value is 30 s (see 8.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTT-Notify(29)};

fddiSMTStatRptPolicy **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.StatRptPolicyType;
BEHAVIOUR
fddiSMTStatRptPolicyBhv BEHAVIOUR
DEFINED AS "If true, indicates that the node will generate Status Reporting Frames for its implemented events and conditions. It has an initial value of true. This attribute determines the value of the SR_Enable Flag (see 8.3.2.1).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTStatRptPolicy(30)};

fddiSMTTrace-MaxExpiration **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.Trace-MaxExpirationType;
BEHAVIOUR
fddiSMTTrace-MaxExpirationBhv BEHAVIOUR
DEFINED AS "Reference Trace_Max (see 9.4.4.2.2).";;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTTrace-MaxExpiration(31)};

fddiSMTPORTIndexes **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.PORTIndexesType;
BEHAVIOUR
fddiSMTPORTIndexesBhv BEHAVIOUR
DEFINED AS "This attribute is an array of indexes of all Port objects within the station.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTPORTIndexes(32)};

fddiSMTMACIndexes **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.MACIndexesType;
BEHAVIOUR
fddiSMTMACIndexesBhv BEHAVIOUR
DEFINED AS "This attribute is an array of indexes of all MAC objects within the station.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTMACIndexes(33)};

fddiSMTBypassPresent **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.BypassPresentType;
BEHAVIOUR
fddiSMTBypassPresentBhv BEHAVIOUR
DEFINED AS "A flag indicating if the station has a bypass on its AB port pair.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTBypassPresent(34)};

fddiSMTECMState **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.ECMStateType;
BEHAVIOUR
fddiSMTECMStateBhv BEHAVIOUR
DEFINED AS "Indicates the current state of the ECM State Machine (see 9.5.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTECMState(41)};

fddiSMTCF-State **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.CF-StateType;
BEHAVIOUR
fddiSMTCF-StateBhv BEHAVIOUR
DEFINED AS "The CF_State represents the attachment configuration for the station or concentrator (see 9.7.2.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTCF-State(42)};

fddiSMTHoldState **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.HoldStateType;
BEHAVIOUR
fddiSMTHoldStateBhv BEHAVIOUR
DEFINED AS "This attribute indicates the current state of the Hold function. The value 'not-holding' is the default and initial value. The value must be set to 'not-holding' as part of Active_Actions and when the conditions causing 'holding-prm' or 'holding-sec' are no longer true. The value 'holding-prm' must be set when the condition (notTR_Flag & not RE_Flag & (PC_LS=QLS | LEM_Fail | TNE>NS_Max | (not LS_Flag & TPC > T_Out))) is satisfied in state PC8:ACTIVE for the A Port. The value 'holding-sec' must be set when the condition (notTR_Flag & not RE_Flag & (PC_LS=QLS | LEM_Fail | TNE>NS_Max | (not LS_Flag & TPC > T_Out))) is satisfied in state PC8:ACTIVE for the B Port.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTHoldState(43)};

fddiSMTRemoteDisconnectFlag **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.RemoteDisconnectFlagType;
BEHAVIOUR
fddiSMTRemoteDisconnectFlagBhv BEHAVIOUR

DEFINED AS "A flag indicating that the station was remotely disconnected from the network as a result of receiving an fddiSMTAction, disconnect (see 6.4.5.3) in a Parameter Management Frame. A station requires a Connect Action to rejoin and clear the flag (see 6.4.5.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTRemoteDisconnectFlag(44)};

fddiSMTStationStatus **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.StationStatusType;
BEHAVIOUR
fddiSMTStationStatusBhv BEHAVIOUR
DEFINED AS "The current status of the primary and secondary paths within this station.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTStationStatus(45)};

fddiSMTPeerWrapFlag **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.PeerWrapFlagType;
BEHAVIOUR
fddiSMTPeerWrapFlagBhv BEHAVIOUR
DEFINED AS "This attribute assumes the value of the PeerWrapFlag in CFM (see 9.7.2.4.4).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTPeerWrapFlag(46)};

fddiSMTTimeStamp **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.TimeStampType;
BEHAVIOUR
fddiSMTTimeStampBhv BEHAVIOUR
DEFINED AS "This attribute assumes the value of TimeStamp (see 8.3.2.1).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTTimeStamp(51)};

fddiSMTTransitionTimeStamp **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.TransitionTimeStampType;
BEHAVIOUR
fddiSMTTransitionTimeStampBhv BEHAVIOUR
DEFINED AS "This attribute assumes the value of TransitionTimeStamp (see 8.3.2.1).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTTransitionTimeStamp(52)};

fddiSMTSetCount **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.SetCountType;
BEHAVIOUR
fddiSMTSetCountBhv BEHAVIOUR
DEFINED AS "This attribute is composed of a count incremented in response to a Set operation on the MIB, and the time of the change (see 8.4.1).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTSetCount(53)};

fddiSMTLastSetStationId **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-SMT.LastSetStationIdType;
BEHAVIOUR
fddiSMTLastSetStationIdBhv BEHAVIOUR
DEFINED AS "The Station ID of the station that effected the last change in the FDDI MIB.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTLastSetStationId(54)};

fddiSMTVendorAttrib **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-COMMON.VendorMIBType;
BEHAVIOUR
fddiSMTVendorAttribBhv BEHAVIOUR

DEFINED AS "This attribute defines a vendor-specified characteristic of this managed object class. This extension is based on the vendor's OUI. This attribute shall not duplicate or redefine any attribute listed in the MIB. ";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16) fddiSMTVendorAttrib(255)};

6.4.5.4.2 MAC Attributes

fddiMACFrameStatusFunctions **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.FrameStatusFunctionsType;
BEHAVIOUR
fddiMACFrameStatusFunctionsBhv BEHAVIOUR
DEFINED AS "Indicates the MAC's optional Frame Status processing functions.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACFrameStatusFunctions(11)};

fddiMACBridgeFunctions **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.BridgeFunctionsType;
BEHAVIOUR
fddiMACBridgeFunctionsBhv BEHAVIOUR
DEFINED AS "Indicates the MAC's optional bridging functions.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACBridgeFunctions(12)};

fddiMACT-MaxCapability **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.T-MaxCapabilityType;
BEHAVIOUR
fddiMACT-MaxCapabilityBhv BEHAVIOUR
DEFINED AS "Indicates the maximum time value (minimum twos-complement numeric value) of fddiMACT-Max that this MAC can support.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-MaxCapability(13)};

fddiMACTVXCapability **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.TVXCapabilityType;
BEHAVIOUR
fddiMACTVXCapabilityBhv BEHAVIOUR
DEFINED AS "Indicates the maximum time value (minimum twos-complement numeric value) of fddiMACTvxValue that this MAC can support.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACTVXCapability(14)};

fddiMACAvailablePaths **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.AvailablePathsType;
BEHAVIOUR
fddiMACAvailablePathsBhv BEHAVIOUR
DEFINED AS "Indicates the paths available for this MAC (see 9.7.7).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACAvailablePaths(22)};

fddiMACCurrentPath **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.CurrentPathType;
BEHAVIOUR
fddiMACCurrentPathBhv BEHAVIOUR
DEFINED AS "Indicates the Path into which this MAC is currently inserted (see 9.7.7).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACCurrentPath(23)};

fddiMACUpstreamNbr **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.UpstreamNbrType;
BEHAVIOUR
fddiMACUpstreamNbrBhv BEHAVIOUR

DEFINED AS "The MAC's upstream neighbour's long individual MAC address. It has an initial value of the SMT-Unknown-MAC Address and is only modified as specified by the Neighbour Information Frame protocol (see 7.2.1 and 8.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACUpstreamNbr(24)};

fddiMACDownstreamNbr ATTRIBUTE
WITH ATTRIBUTE SYNTAX FDDI-MAC.DownstreamNbrType;
BEHAVIOUR
fddiMACDownstreamNbrBhv BEHAVIOUR
DEFINED AS "The MAC's downstream neighbour's long individual MAC address. It has an initial value of the SMT-Unknown-MAC Address and is only modified as specified by the Neighbour Information Frame protocol (see 7.2.1 and 8.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACDownstreamNbr(25)};

fddiMACOldUpstreamNbr ATTRIBUTE
WITH ATTRIBUTE SYNTAX FDDI-MAC.OldUpstreamNbrType;
BEHAVIOUR
fddiMACOldUpstreamNbrBhv BEHAVIOUR
DEFINED AS "The previous value of the MAC's upstream neighbour's long individual MAC address. It has an initial value of the SMT-Unknown-MAC Address and is only modified as specified by the Neighbour Information Frame protocol (see 7.2.1 and 8.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACOldUpstreamNbr(26)};

fddiMACOldDownstreamNbr ATTRIBUTE
WITH ATTRIBUTE SYNTAX FDDI-MAC.OldDownstreamNbrType;
BEHAVIOUR
fddiMACOldDownstreamNbrBhv BEHAVIOUR
DEFINED AS "The previous value of the MAC's downstream neighbour's long individual MAC address. It has an initial value of the SMT-Unknown-MAC Address and is only modified as specified by the Neighbour Information Frame protocol (see 7.2.1 and 8.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACOldDownstreamNbr(27)};

fddiMACDupAddressTest ATTRIBUTE
WITH ATTRIBUTE SYNTAX FDDI-MAC.DupAddressTestType;
BEHAVIOUR
fddiMACDupAddressTestBhv BEHAVIOUR
DEFINED AS "The Duplicate Address Test flag, Dup_Addr_Test (see 8.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACDupAddressTest(29)};

fddiMACRequestedPaths ATTRIBUTE
WITH ATTRIBUTE SYNTAX FDDI-MAC.RequestedPathsType;
BEHAVIOUR
fddiMACRequestedPathsBhv BEHAVIOUR
DEFINED AS "List of permitted Paths which specifies the Path(s) into which the MAC may be inserted (see 9.7).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACRequestedPaths(32)};

fddiMACDownstreamPORTType ATTRIBUTE
WITH ATTRIBUTE SYNTAX FDDI-MAC.DownstreamPORTType;
BEHAVIOUR
fddiMACDownstreamPORTTypeBhv BEHAVIOUR
DEFINED AS "Indicates the PC_Type of the first port that is downstream of this MAC (the exit port).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACDownstreamPORTType(33)};

fddiMACIndex **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.MACIndexType;
BEHAVIOUR
fddiMACIndexBhv BEHAVIOUR
DEFINED AS "Index attribute for uniquely identifying the MAC object instances.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACIndex(34)};

fddiMACSMTAddress **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.SMTAddressType;
BEHAVIOUR
fddiMACSMTAddressBhv BEHAVIOUR
DEFINED AS "The 48-bit individual address of the MAC used for SMT frames.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32)
fddiMACSMTAddress(41)};

fddiMACLongGrpAddress **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.LongGrpAddressType;
BEHAVIOUR
fddiMACLongGrpAddressBhv BEHAVIOUR
DEFINED AS "A possibly null set of 48-bit group addresses for this MAC.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32)
fddiMACLongGrpAddress(44)};

fddiMACShortGrpAddress **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.ShortGrpAddressType;
BEHAVIOUR
fddiMACShortGrpAddressBhv BEHAVIOUR
DEFINED AS "A possibly null set of 16-bit group addresses for this MAC.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32)
fddiMACShortGrpAddress(45)};

fddiMACT-Req **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.TReqType;
BEHAVIOUR
fddiMACT-ReqBhv BEHAVIOUR
DEFINED AS "This attribute is the T_Req_value passed to the MAC. It is reported as a twos-complement number as described in the MAC standard. Without having detected a duplicate, the time value of this attribute shall assume the maximum supported time value (minimum supported twos-complement value) which is less than or equal to the time value of fddiPATHMaxT-Req. When a MAC has an address detected as a duplicate, it may use a time value for this attribute greater than the time value of fddiPATHT-MaxLowerBound. A station shall cause claim when the new T_Req may cause the value of T_Neg to change in the claim process, (i.e., time value new T_Req < T_Neg, or old T_Req = T_Neg).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-Req(51)};

fddiMACT-Neg **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.TNegType;
BEHAVIOUR
fddiMACT-NegBhv BEHAVIOUR
DEFINED AS "It is reported as a twos-complement number as described in the MAC standard.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-Neg(52)};

fddiMACT-Max **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.TMaxType;
BEHAVIOUR
fddiMACT-MaxBhv BEHAVIOUR
DEFINED AS "This attribute is the T_Max_value passed to the MAC. It is reported as a twos-complement number as described in the MAC standard. The time value of this

attribute shall assume the minimum supported time value (maximum supported twos-complement value) which is greater than or equal to the time value of fddiPATHT-MaxLowerBound.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-Max(53)};

fddiMACTvxValue **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.TvxValueType;
BEHAVIOUR
fddiMACTvxValueBhv BEHAVIOUR
DEFINED AS "This attribute is the TVX_value passed to the MAC. It is reported as a twos-complement number as described in the MAC standard. The time value of this attribute shall assume the minimum supported time value (maximum supported twos-complement value) which is greater than or equal to the time value of fddiPATHTVXLowerBound.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACTvxValue(54)};

fddiMACT-Pri0 **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.T-Pri0Type;
BEHAVIOUR
fddiMACT-Pri0Bhv BEHAVIOUR
DEFINED AS "This attribute is an unsigned twos-complement T_pri threshold as described in the MAC standard.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-Pri0(56)};

fddiMACT-Pri1 **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.T-Pri1Type;
BEHAVIOUR
fddiMACT-Pri1Bhv BEHAVIOUR
DEFINED AS "This attribute is an unsigned twos-complement T_pri threshold as described in the MAC standard.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-Pri1(57)};

fddiMACT-Pri2 **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.T-Pri2Type;
BEHAVIOUR
fddiMACT-Pri2Bhv BEHAVIOUR
DEFINED AS "This attribute is an unsigned twos-complement T_pri threshold as described in the MAC standard.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-Pri2(58)};

fddiMACT-Pri3 **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.T-Pri3Type;
BEHAVIOUR
fddiMACT-Pri3Bhv BEHAVIOUR
DEFINED AS "This attribute is an unsigned twos-complement T_pri threshold as described in the MAC standard.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-Pri3(59)};

fddiMACT-Pri4 **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.T-Pri4Type;
BEHAVIOUR
fddiMACT-Pri4Bhv BEHAVIOUR
DEFINED AS "This attribute is an unsigned twos-complement T_pri threshold as described in the MAC standard.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-Pri4(60)};

fddiMACT-Pri5 **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.T-Pri5Type;
BEHAVIOUR
fddiMACT-Pri5Bhv BEHAVIOUR
DEFINED AS "This attribute is an unsigned twos-complement T_pri threshold as described in the MAC standard.";;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-Pri5(61)};

fddiMACT-Pri6 **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.T-Pri6Type;
BEHAVIOUR
fddiMACT-Pri6Bhv BEHAVIOUR
DEFINED AS "This attribute is an unsigned twos-complement T_pri threshold as described in the MAC standard.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACT-Pri6(62)};

fddiMACFrame-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.Frame-CtType;
BEHAVIOUR
fddiMACFrame-CtBhv BEHAVIOUR
DEFINED AS "A count of the number of frames received by this MAC (see MAC 7.5.1).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACFrame-Ct(71)};

fddiMACCopied-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.Copied-CtType;
BEHAVIOUR
fddiMACCopied-CtBhv BEHAVIOUR
DEFINED AS "A count that should as closely as possible match the number of frames addressed to (A bit set) and successfully copied into the station's receive buffers (C bit set) by this MAC (see MAC 7.5). Note that this count does not include MAC frames.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACCopied-Ct(72)};

fddiMACTransmit-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.Transmit-CtType;
BEHAVIOUR
fddiMACTransmit-CtBhv BEHAVIOUR
DEFINED AS "A count that should as closely as possible match the number of frames transmitted by this MAC (see MAC 7.5). Note that this count does not include MAC frames.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACTransmit-Ct(73)};

fddiMACToken-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.Token-CtType;
BEHAVIOUR
fddiMACToken-CtBhv BEHAVIOUR
DEFINED AS "A count that should as closely as possible match the number of times the station has received a token (total of non-restricted and restricted) on this MAC (see MAC 7.4). This count is valuable for determination of network load.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACToken-Ct(74)};

fddiMACError-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.Error-CtType;
BEHAVIOUR
fddiMACError-CtBhv BEHAVIOUR
DEFINED AS "A count of the number of frames that were detected in error by this MAC that had not been detected in error by another MAC (see MAC 7.5.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACError-Ct(81)};

fddiMACLost-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.Lost-CtType;
BEHAVIOUR
fddiMACLost-CtBhv BEHAVIOUR
DEFINED AS "A count of the number of instances that this MAC detected a format error during frame reception such that the frame was stripped (see MAC 7.5.3).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACLost-Ct(82)};

fddiMACTvxExpired-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.TvxExpired-CtType;
BEHAVIOUR
fddiMACTvxExpired-CtBhv BEHAVIOUR
DEFINED AS "A count that should as closely as possible match the number of times that TVX has expired.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACTvxExpired-Ct(83)};

fddiMACNotCopied-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.NotCopied-CtType;
BEHAVIOUR
fddiMACNotCopied-CtBhv BEHAVIOUR
DEFINED AS "A count that should as closely as possible match the number of frames that were addressed to this MAC but were not copied into its receive buffers (see MAC 7.5). For example, this might occur due to local buffer congestion. Because of implementation considerations, this count may not match the actual number of frames not copied. It is not a requirement that this count be exact. Note that this count does not include MAC frames.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACNotCopied-Ct(84)};

fddiMACLate-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.Late-CtType;
BEHAVIOUR
fddiMACLate-CtBhv BEHAVIOUR
DEFINED AS "A count that should as closely as possible match the number of TRT expirations since this MAC was reset or a token was received (see MAC 7.4.5).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACLate-Ct(85)};

fddiMACRingOp-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.RingOp-CtType;
BEHAVIOUR
fddiMACRingOp-CtBhv BEHAVIOUR
DEFINED AS "The count of the number of times the ring has entered the 'Ring_Operational' state from the 'Ring Not Operational' state. This count is updated when a SM_MA_STATUS.Indication of a change in the Ring_Operational status occurs (see 6.1.4). Because of implementation considerations, this count may be less than the actual RingOp_Ct. It is not a requirement that this count be exact.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACRingOp-Ct(86)};

fddiMACFrameErrorThreshold **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.FrameErrorThresholdType;
BEHAVIOUR
fddiMACFrameErrorThresholdBhv BEHAVIOUR
DEFINED AS "A threshold for determining when a MAC Condition report (see 8.3.1.1) shall be generated. Stations not supporting variable thresholds shall have a value of 0 and a range of (0..0).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACFrameErrorThreshold(95)};

fddiMACFrameErrorRatio **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.FrameErrorRatioType;
BEHAVIOUR
fddiMACFrameErrorRatioBhv BEHAVIOUR
DEFINED AS "This attribute is the value of the ratio,

Erreur!• 2^{16} ";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACFrameErrorRatio(96)};

fddiMACNotCopiedThreshold **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.NotCopiedThresholdType;
BEHAVIOUR
fddiMACNotCopiedThresholdBhv BEHAVIOUR
DEFINED AS "A threshold for determining when a MAC Condition report (see 8.3.1.1) shall be generated. Stations not supporting variable thresholds shall have a value of 0 and a range of (0..0).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACNotCopiedThreshold(103)};

fddiMACNotCopiedRatio **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.NotCopiedRatioType;
BEHAVIOUR
fddiMACNotCopiedRatioBhv BEHAVIOUR
DEFINED AS "This attribute is the value of the ratio:
Erreur!• 2^{16} ";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACNotCopiedRatio(105)};

fddiMACRMTState **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.RMTStateType;
BEHAVIOUR
fddiMACRMTStateBhv BEHAVIOUR
DEFINED AS "Indicates the current state of the RMT State Machine (see 10.3.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACRMTState(111)};

fddiMACDA-Flag **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.DA-FlagType;
BEHAVIOUR
fddiMACDA-FlagBhv BEHAVIOUR
DEFINED AS "The RMT Duplicate Address Flag, DA_Flag (see 10.2.1.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACDA-Flag(112)};

fddiMACUNDA-Flag **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.UNDA-FlagType;
BEHAVIOUR
fddiMACUNDA-FlagBhv BEHAVIOUR
DEFINED AS "A flag, UNDA_Flag (see 8.2.2.1), set when the upstream neighbour reports a duplicate address condition. Cleared when the condition clears.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACUNDA-Flag(113)};

fddiMACFrameErrorFlag **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.FrameErrorFlagType;
BEHAVIOUR
fddiMACFrameErrorFlagBhv BEHAVIOUR
DEFINED AS "Indicates the MAC Frame Error Condition is present when set. Cleared when the condition clears and on station initialization.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACFrameErrorFlag(114)};

fddiMACNotCopiedFlag **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.NotCopiedFlagType;
BEHAVIOUR
fddiMACNotCopiedFlagBhv BEHAVIOUR
DEFINED AS "Indicates the Not Copied Condition is present when set. Cleared when the condition clears and on station initialization.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACNotCopiedFlag(115)};

fddiMACMA-UnitdataAvailable **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.MA-UnitdataAvailableType;
BEHAVIOUR
fddiMACMA-UnitdataAvailableBhv BEHAVIOUR
DEFINED AS "This attribute shall take on the value of the MAC_Avail flag defined in RMT.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACMA-UnitdataAvailable(116)};

fddiMACHardwarePresent **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.HWPresentType;
BEHAVIOUR
fddiMACHardwarePresentBhv BEHAVIOUR
DEFINED AS "This attribute indicates the presence of underlying hardware support for this MAC object.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACHardwarePresent(117)};

fddiMACMA-UnitdataEnable **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-MAC.MA-UnitdataEnableType;
BEHAVIOUR
fddiMACMA-UnitdataEnableBhv BEHAVIOUR
DEFINED AS "This attribute determines the value of the MA_UNITDATA_Enable flag in RMT. The default and initial value of this flag is set.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACMA-UnitdataEnable(118)};

fddiMACVendorAttrib **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-COMMON.VendorMIBType;
BEHAVIOUR
fddiMACVendorAttribBhv BEHAVIOUR
DEFINED AS "This attribute defines a vendor-specified characteristic of this managed object class. This extension is based on the vendor's OUI. This attribute shall not duplicate or redefine any attribute listed in the MIB.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACVendorAttrib(255)};

6.4.5.4.3 PATH attributes

fddiPATHIndex **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PATH.IndexType;
BEHAVIOUR
fddiPATHIndexBhv BEHAVIOUR
DEFINED AS "Index attribute for uniquely identifying the primary, secondary and local PATH object instances. Local PATH object instances are represented with integer values 3 to 255.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHIndex(11)};

fddiPATHRingLatency **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PATH.RingLatencyType;
BEHAVIOUR
fddiPATHRingLatencyBhv BEHAVIOUR
DEFINED AS "Gives the total accumulated latency of the ring associated with this path. May be measured directly by the station or may be calculated by a management station.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHRingLatency(13)};

fddiPATHTraceStatus **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PATH.TraceStatusType;
BEHAVIOUR
fddiPATHTraceStatusBhv BEHAVIOUR

DEFINED AS "This attribute indicates the current trace status of the path (see 9.5.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHTraceStatus(14)};

fddiPATHSbaPayload **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PATH.SbaPayloadType;
BEHAVIOUR
fddiPATHSbaPayloadBhv BEHAVIOUR
DEFINED AS "The payload portion of the Synchronous Bandwidth Allocation for this path. This value represents the maximum number of bytes of data allocated for transmission per 125 microseconds.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHSbaPayload(15)};

fddiPATHSbaOverhead **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PATH.SbaOverheadType;
BEHAVIOUR
fddiPATHSbaOverheadBhv BEHAVIOUR
DEFINED AS "The overhead portion of the Synchronous Bandwidth Allocation for this path. This value represents the maximum number of bytes of overhead (token capture, frame header, etc.) used per negotiated Target Token Rotation Time (T_Neg).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHSbaOverhead(16)};

fddiPATHConfiguration **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PATH.ConfigurationType;
BEHAVIOUR
fddiPATHConfigurationBhv BEHAVIOUR
DEFINED AS "A circular list of resources on the path where each entry in the list consists of a resource type identifier (MAC or PORT), an index, and its CurrentPath. If the path is available to a resource in the node (as indicated by fddiPORTAvailablePaths or fddiMACAvailablePaths) then that resource shall be included in the list. If the path is not available to a resource or the underlying hardware is not present, then that resource shall not be included in the list. Resources currently inserted in the Path shall appear in the list once in token order. Resources not currently inserted in the Path shall appear once in any position supported by the implementation.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHConfiguration(18)};

fddiPATHT-Rmode **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PATH.T-RmodeType;
BEHAVIOUR
fddiPATHT-RmodeBhv BEHAVIOUR
DEFINED AS "Used by RMT to limit the duration of restricted dialogs on a path.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHT-Rmode(19)};

fddiPATHSbaAvailable **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PATH.SbaAvailableType;
BEHAVIOUR
fddiPATHSbaAvailableBhv BEHAVIOUR
DEFINED AS "This value is the maximum Synchronous Bandwidth available for a path in bytes per second.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHSbaAvailable(20)};

fddiPATHTVXLowerBound **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PATH.TVXLowerBoundType;
BEHAVIOUR
fddiPATHTVXLowerBoundBhv BEHAVIOUR

DEFINED AS "Specifies the minimum time value (maximum twos-complement numeric value) of fddiMACTvxValue that shall be used by any MAC that is configured in this path. The operational value of fddiMACTvxValue is managed by settting this attribute. This attribute has the time value range of:

0 < fddiPATHTVXLowerBound < fddiPATHMaxT-Req.

Changes to this attribute shall either satisfy the time value relationship:

fddiPATHTVXLowerBound ² fddiMACTVXCapability

of each of the MACs currently on the path, or be considered out of range. The initial value of fddiPATHTVXLowerBound shall be 2.500 msec.";;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHTVXLowerBound(21)};

fddiPATHT-MaxLowerBound **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX FDDI-PATH.T-MaxLowerBoundType;

BEHAVIOUR

fddiPATHT-MaxLowerBoundBhv BEHAVIOUR

DEFINED AS "Specifies the minimum time value (maximum twos-complement numeric value) of fddiMACT-Max that shall be used by any MAC that is configured in this path. The operational value of fddiMACT-Max is managed by setting this attribute. This attribute has the time value range of:

fddiPATHMaxT-Req ² fddiPATHT-MaxLowerBound

and an absolute time value range of

10 msec. ² fddiPATHT-MaxLowerBound.

Changes to this attribute shall either satisfy the time value relationship:

fddiPATHT-MaxLowerBound < fddiMACT-MaxCapability

of each of the MACs currently on the path, or be considered out of range. The initial value of fddiPATHT-MaxLowerBound shall be 165 msec.";;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHT-MaxLowerBound(22)};

fddiPATHMaxT-Req **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX FDDI-PATH.MaxT-ReqType;

BEHAVIOUR

fddiPATHMaxT-ReqBhv BEHAVIOUR

DEFINED AS "Specifies the maximum time value (minimum twos-complement numeric value) of fddiMACT-Req that shall be used by any MAC that is configured in this path. The operational value of fddiMACT-Req is managed by setting this attribute. This attribute has the time value range of:

fddiPATHTVXLowerBound < fddiPATHMaxT-Req ²

fddiPATHT-MaxLowerBound.

The default value of fddiPATHMaxT-Req is 165 msec.";;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHMaxT-Req(23)};

fddiPATHVendorAttrib **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX FDDI-COMMON.VendorMIBType;

BEHAVIOUR

fddiPATHVendorAttribBhv BEHAVIOUR

DEFINED AS "This attribute defines a vendor-specified characteristic of this managed object class. This extension is based on the vendor's OUI. This attribute shall not duplicate or redefine any attribute listed in the MIB.";;

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHVendorAttrib(255)};

6.4.5.4.4 PORT attributes

fddiPORTMy-Type **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX FDDI-PORT.MyTypeType;

BEHAVIOUR

fddiPORTMy-TypeBhv BEHAVIOUR

DEFINED AS "The value of the PORT's PC_Type (see 9.4.1, and 9.6.3.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTMy-Type(12)};

fddiPORTNeighbourType **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.NeighbourTypeType;
BEHAVIOUR
fddiPORTNeighbourTypeBhv BEHAVIOUR
DEFINED AS "The type of the remote PORT as determined in PCM. This attribute has an initial value of none, and is only modified in PC_RCode(3)_Actions (see 9.6.3.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTNeighbourType(13)};

fddiPORTConnectionPolicies **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.ConnectionPoliciesType;
BEHAVIOUR
fddiPORTConnectionPoliciesBhv BEHAVIOUR
DEFINED AS "A bit string representing the PORT's connection policies desired in the node. The value of pc-mac-lct is a term used in the PC_MAC_LCT Flag (see 9.4.3.2). The value of pc-mac-loop is a term used in the PC_MAC_Loop Flag.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTConnectionPolicies(14)};

fddiPORTMACIndicated **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.MACIndicatedType;
BEHAVIOUR
fddiPORTMACIndicatedBhv BEHAVIOUR
DEFINED AS "The indications (T_Val(9), R_Val(9)) in PC-Signalling, of the intent to place a MAC in the output token path to a PORT (see 9.6.3.2.).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTMACIndicated(15)};

fddiPORTCurrentPath **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.CurrentPathType;
BEHAVIOUR
BfddiPORTCurrentPathhv BEHAVIOUR
DEFINED AS "Indicates the Path(s) into which this PORT is currently inserted.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTCurrentPath(16)};

fddiPORTRequestedPaths **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.RequestedPathsType;
BEHAVIOUR
fddiPORTRequestedPathsBhv BEHAVIOUR
DEFINED AS "This attribute is a list of permitted Paths where each list element defines the Port's permitted Paths (see 9.7).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTRequestedPaths(17)};

fddiPORTMACPlacement **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.MACPlacementType;
BEHAVIOUR
fddiPORTMACPlacementBhv BEHAVIOUR
DEFINED AS "Indicates the MAC, if any, whose transmit path exits the station via this PORT. The value shall be zero if there is no MAC associated with the PORT. Otherwise, the MACIndex of the MAC will be the value of the attribute.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTMACPlacement(18)};

fddiPORTAvailablePaths **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.AvailablePathsType;

```
    BEHAVIOUR
        fddiPORTAvailablePathsBhv BEHAVIOUR
            DEFINED AS    "Indicates the Paths which are available to this Port. In the absence of
            faults, the A and B Ports will always have both the Primary and Secondary Paths
            available.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64)
                fddiPORTAvailablePaths(19)};

fddiPORTMACLoop-Time                                ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-PORT.MACLoop-TimeType;
    BEHAVIOUR
        fddiPORTMACLoopBhv BEHAVIOUR
            DEFINED AS    "This attribute controls the value used for T_Next(9) (see 9.4.4.2.3).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTMACLoop-
                Time(21)};

fddiPORTPMDClass                                    ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-PORT.PMDClassType;
    BEHAVIOUR
        fddiPORTPMDClassBhv BEHAVIOUR
            DEFINED AS    "This attribute indicates the type of PMD entity associated with this port.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64)
                fddiPORTPMDClass(22)};

fddiPORTConnectionCapabilities                      ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-PORT.ConnectionCapabilitiesType;
    BEHAVIOUR
        fddiPORTConnectionCapabilitiesBhv BEHAVIOUR
            DEFINED AS    "A bit string that indicates the connection capabilities of the port. The pc-
            mac-lct bit indicates that the station has the capability of setting the PC_MAC_LCT Flag.
            The pc-mac-loop bit indicates that the station has the capability of setting the
            PC_MAC_Loop Flag (see 9.4.3.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64)
                fddiPORTConnectionCapabilities(23)};

fddiPORTIndex                                       ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-PORT.PortIndexType;
    BEHAVIOUR
        fddiPORTIndexBhv BEHAVIOUR
            DEFINED AS    "Index attribute for uniquely identifying the PORT object instances.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTIndex(29)};

fddiPORTMaint-LS                                    ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-PORT.Maint-LSType;
    BEHAVIOUR
        fddiPORTMaint-LSBhv BEHAVIOUR
            DEFINED AS    "The PORT Maintenance Line State specifies the line state (Maint_LS) to
            be transmitted when the PCM state machine for the port is in state PC9 Maint.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTMaint-
                LS(31)};

fddiPORTBS-Flag                                     ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-PORT.BS-FlagType;
    BEHAVIOUR
        fddiPORTBS-FlagBhv BEHAVIOUR
            DEFINED AS    "This attribute assumes the value of the BS_Flag (see 9.4.3.3).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTBS-Flag(33)};

fddiPORTPC-LS                                       ATTRIBUTE
    WITH ATTRIBUTE SYNTAX          FDDI-PORT.PC-LSType;
    BEHAVIOUR
        fddiPORTPC-LSBhv BEHAVIOUR
```

DEFINED AS "This attribute indicates the line state (PC_LS) received by the port.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTPC-LS(34)};

fddiPORTEBError-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.EBError-CtType;
BEHAVIOUR
fddiPORTEBError-CtBhv BEHAVIOUR
DEFINED AS "A count that should as closely as possible match the times an Elasticity Buffer Error has occurred while in active line state.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTEBError-Ct(41)};

fddiPORTLCTFail-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.LCTFail-CtType;
BEHAVIOUR
fddiPORTLCTFail-CtBhv BEHAVIOUR
DEFINED AS "The count of the consecutive times the link confidence test (LCT) has failed during connection management (see 9.4.1).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTLCTFail-Ct(42)};

fddiPORTLer-Estimate **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.Ler-EstimateType;
BEHAVIOUR
fddiPORTLer-EstimateBhv BEHAVIOUR
DEFINED AS "A long-term average link error rate. It ranges from 10(-4) to 10(-15) and is reported as the absolute value of the base 10 logarithm (see 9.4.7.5.).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTLer-Estimate(51)};

fddiPORTLem-Reject-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.Lem-Reject-CtType;
BEHAVIOUR
fddiPORTLem-Reject-CtBhv BEHAVIOUR
DEFINED AS "A link error monitoring count of the times that a link has been rejected.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTLem-Reject-Ct(52)};

fddiPORTLem-Ct **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.Lem-CtType;
BEHAVIOUR
fddiPORTLem-CtBhv BEHAVIOUR
DEFINED AS "The aggregate link error monitor error count, set to zero only on station initialization.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTLem-Ct(53)};

fddiPORTLer-Cutoff **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.Ler-CutoffType;
BEHAVIOUR
fddiPORTLer-CutoffBhv BEHAVIOUR
DEFINED AS "The link error rate estimate at which a link connection will be broken. It ranges from 10(-4) to 10(-15) and is reported as the absolute value of the base 10 logarithm (default of 7).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTLer-Cutoff(58)};

fddiPORTLer-Alarm **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.Ler-AlarmType;
BEHAVIOUR
fddiPORTLer-AlarmBhv BEHAVIOUR

DEFINED AS "The link error rate estimate at which a link connection will generate an alarm. It ranges from 10(-4) to 10(-15) and is reported as the absolute value of the base 10 logarithm of the estimate (default of 8).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTLer-Alarm(59)};

fddiPORTConnectState **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.ConnectStateType;
BEHAVIOUR
fddiPORTConnectStateBhv BEHAVIOUR
DEFINED AS "An indication of the connect state of this PORT and is equal to the value of Connect_State (see 9.4.1).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTConnectState(61)};

fddiPORTPCMState **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.PCMStateType;
BEHAVIOUR
fddiPORTPCMStateBhv BEHAVIOUR
DEFINED AS "The state of this Port's PCM state machine (see 9.6.2).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTPCMState(62)};

fddiPORTPC-Withhold **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.PC-WithholdType;
BEHAVIOUR
fddiPORTPC-WithholdBhv BEHAVIOUR
DEFINED AS "The value of PC_Withhold (see 9.4.1).";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTPC-Withhold(63)};

fddiPORTLerFlag **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.LerFlagType;
BEHAVIOUR
fddiPORTLerFlagBhv BEHAVIOUR
DEFINED AS "This attribute is set to TRUE whenever fddiPORTLerEstimate is less than or equal to fddiPORTLerAlarm.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTLerFlag64)};

fddiPORTHardwarePresent **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-PORT.HWPresentType;
BEHAVIOUR
fddiPORTHardwarePresentBhv BEHAVIOUR
DEFINED AS "This attribute indicates the presence of underlying hardware support for this port object.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTHardwarePresent(65)};

fddiPORTVendorAttrib **ATTRIBUTE**
WITH ATTRIBUTE SYNTAX FDDI-COMMON.VendorMIBType;
BEHAVIOUR
fddiPORTVendorAttribBhv BEHAVIOUR
DEFINED AS "This attribute defines a vendor-specified characteristic of this managed object class. This extension is based on the vendor's OUI. This attribute shall not duplicate or redefine any attribute listed in the MIB.";;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTVendorAttrib(255)};

6.4.5.5 Action templates

6.4.5.5.1 SMT actions

```
fddiSMTStationAction                                    ACTION
    BEHAVIOUR
        fddiSMTStationActionBhv BEHAVIOUR
            DEFINED AS "The behaviour of these actions is the following:
                Connect:        Generates a Connect signal to ECM to begin a connection sequence.
                                The fddiSMTRemoteDisconnectFlag is cleared on this action (see
                                9.4.2).
                Disconnect:     Generates a Disconnect signal to ECM and sets the
                                fddiSMTRemoteDIsconnectFlag if received in a Parameter
                                Management Frame (see 9.4.2).
                Path_Test:      Initiates a station Path_Test. The Path_Test variable (see 9.4.1) is set
                                to 'Testing'. The results of this action are not specified in this standard.
                Self_Test:      Initiates a station Self_Test. The results of this action are not specified
                                in this standard.
                Disable_A       Causes a PC_Disable on the A port if the A port mode is peer.
                Disable_B       Causes a PC_Disable on the B port if the B port mode is peer.
                Disable_M       Causes a PC_Disable on all M ports.";;
    WITH INFORMATION SYNTAX     FDDI-SMT.ActionInfoType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)
                    fddiSMTStationAction(60)};

fddiSMTVendorAction                                     ACTION
    BEHAVIOUR
        fddiSMTVendorActionBhv BEHAVIOUR
            DEFINED AS "This action is defined by the vendor. It provides the capability to define a
            vendor-specific action based on the manufacturer's OUI. This action shall not duplicate or
            redefine any action listed in the MIB.";;
    WITH INFORMATION SYNTAX     FDDI-COMMON.VendorMIBType;
    WITH REPLY SYNTAX           FDDI-COMMON.VendorMIBType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)
                    fddiSMTVendorAction(254)};
```

6.4.5.5.2 MAC actions

```
fddiMACVendorAction                                     ACTION
    BEHAVIOUR
        fddiMACVendorActionBhv BEHAVIOUR
            DEFINED AS "This action is defined by the vendor. It provides the capability to define a
            vendor-specific action based on the manufacturer's OUI. This action shall not duplicate or
            redefine any action listed in the MIB.";;
    WITH INFORMATION SYNTAX     FDDI-COMMON.VendorMIBType;
    WITH REPLY SYNTAX           FDDI-COMMON.VendorMIBType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32)
                    fddiMACVendorAction(254)};
```

6.4.5.5.3 PATH actions

```
fddiPATHVendorAction                                    ACTION
    BEHAVIOUR
        fddiPATHVendorActionBhv BEHAVIOUR
            DEFINED AS "This action is defined by the vendor. It provides the capability to define a
            vendor-specific action based on the manufacturer's OUI. This action shall not duplicate or
            redefine any action listed in the MIB.";;
    WITH INFORMATION SYNTAX     FDDI-COMMON.VendorMIBType;
    WITH REPLY SYNTAX           FDDI-COMMON.VendorMIBType;
```

REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50)
fddiPATHVendorAction(254)};

6.4.5.5.4 PORT actions

fddiPORTAction **ACTION**
BEHAVIOUR
fddiPORTActionBhv BEHAVIOUR
DEFINED AS "Causes a Control signal to be generated with a control_action of 'Signal' and the 'variable' parameter set with the appropriate value (i.e., PC_Maint, PC_Enable, PC_Disable, PC_Start, or PC_Stop) (see 9.4.2).";;
WITH INFORMATION SYNTAX FDDI-PORT.ActionInfoType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTAction(70)};

fddiPORTVendorAction **ACTION**
BEHAVIOUR
fddiPORTVendorActionBhv BEHAVIOUR
DEFINED AS "This action is defined by the vendor. It provides the capability to define a vendor-specific action based on the manufacturer's OUI. This action shall not duplicate or redefine any action listed in the MIB.";;
WITH INFORMATION SYNTAX FDDI-COMMON.VendorMIBType;
WITH REPLY SYNTAX FDDI-COMMON.VendorMIBType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64)
fddiPORTVendorAction(254)};

6.4.5.6 Notification templates

6.4.5.6.1 SMT notifications

fddiSMTHoldCondition **NOTIFICATION**
BEHAVIOUR
fddiSMTHoldConditionBhv BEHAVIOUR
DEFINED AS "Generated when fddiSMTHoldState assumes the state holding-prm or holding-sec. This notification is a Condition in the Status Report Protocol (see 7.2.7 and 8.3).";;
WITH INFORMATION SYNTAX FDDI-SMT.HoldConditionDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)
fddiSMTHoldCondition(71)};

fddiSMTPeerWrapCondition **NOTIFICATION**
BEHAVIOUR
fddiSMTPeerWrapConditionBhv BEHAVIOUR
DEFINED AS "This condition is active when fddiSMTPeerWrapFlag is set. This notification is a Condition in the Status Report Protocol (see 7.2.7 and 8.3).";;
WITH INFORMATION SYNTAX FDDI-SMT.PeerWrapConditionDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)
fddiSMTPeerWrapCondition(72)};

fddiSMTVendorNotification **NOTIFICATION**
BEHAVIOUR
fddiSMTVendorNotificationBhv BEHAVIOUR
DEFINED AS "This notification is defined by the vendor. It provides the capability to define a vendor-specific notification based on the manufacturer's OUI. This action shall not duplicate or redefine any notification listed in the MIB.";;
WITH INFORMATION SYNTAX SET OF FDDI-COMMON.VendorMIBType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiSMT(16)
fddiSMTVendorNotification(252)};

6.4.5.6.2 MAC notifications

fddiMACDuplicateAddressCondition **NOTIFICATION**
BEHAVIOUR
fddiMACDuplicateAddressConditionBhv BEHAVIOUR
DEFINED AS "This condition is active when either fddiMACDA-Flag or fddiMACUNDA-Flag is set. This notification is handled as a Condition in the Status Report Protocol (see 7.2.7 and 8.3).";;
WITH INFORMATION SYNTAX FDDI-MAC.DuplicateAddressDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32)fddiMACDuplicateAddressCondition(140)};

fddiMACFrameErrorCondition **NOTIFICATION**
BEHAVIOUR
fddiMACFrameErrorConditionBhv BEHAVIOUR
DEFINED AS "Generated when the fddiMACFrameErrorRatio is greater than or equal to fddiMACFrameErrorThreshold. This notification is handled as a Condition in the Status Report Protocol (see 7.2.7 and 8.3).";;
WITH INFORMATION SYNTAX FDDI-MAC.FrameErrorDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACFrameErrorCondition(141)};

fddiMACNotCopiedCondition **NOTIFICATION**
BEHAVIOUR
fddiMACNotCopiedConditionBhv BEHAVIOUR
DEFINED AS "Generated when the fddiMACNotCopiedRatio is greater than or equal fddiMACNotCopiedThreshold. This notification is handled as a Condition in the Status Report Protocol (see 7.2.7 and 8.3).";;
WITH INFORMATION SYNTAX FDDI-MAC.NotCopiedDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACNotCopiedCondition(142)};

fddiMACNeighbourChangeEvent **NOTIFICATION**
BEHAVIOUR
fddiMACNeighbourChangeEventBhv BEHAVIOUR
DEFINED AS "Generated when a change in a MAC's upstream neighbour address or downstream neighbour address is detected (see 7.2.7 and 8.3).";;
WITH INFORMATION SYNTAX FDDI-MAC.NeighbourChangeDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACNeighbourChangeEvent(143)};

fddiMACPathChangeEvent **NOTIFICATION**
BEHAVIOUR
fddiMACPathChangeEventBhv BEHAVIOUR
DEFINED AS "This event is generated when the value of the fddiMACCurrentPath changes. This event shall be suppressed if the value changes from isolated to local or local to isolated (see 7.2.7 and 8.3).";;
WITH INFORMATION SYNTAX FDDI-MAC.PathChangeDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACPathChangeEvent(144)};

fddiMACVendorNotification **NOTIFICATION**
BEHAVIOUR
fddiMACVendorNotificationBhv BEHAVIOUR
DEFINED AS "This notification is defined by the vendor. It provides the capability to define a vendor-specific notification based on the manufacturer's OUI. This action shall not duplicate or redefine any notification listed in the MIB.";;
WITH INFORMATION SYNTAX SET OF FDDI-COMMON.VendorMIBType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiMAC(32) fddiMACVendorNotification(252)};

6.4.5.6.3 PATH notifications

fddiPATHVendorNotification **NOTIFICATION**
BEHAVIOUR
fddiPATHVendorNotificationBhv BEHAVIOUR
DEFINED AS "This notification is defined by the vendor. It provides the capability to define a vendor-specific notification based on the manufacturer's OUI. This action shall not duplicate or redefine any notification listed in the MIB.";;
WITH INFORMATION SYNTAX SET OF FDDI-COMMON.VendorMIBType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPATH(50) fddiPATHVendorNotification(252)};

6.4.5.6.4 PORT notifications

fddiPORTLerCondition **NOTIFICATION**
BEHAVIOUR
fddiPORTLerConditionBhv BEHAVIOUR
DEFINED AS "The condition becomes active when the value of fddiPORTLer-Estimate is less than or equal to fddiPORTLer-Alarm. This will be reported with the Status Report Frames (SRF) (see 7.2.7 and 8.3).";;
WITH INFORMATION SYNTAX FDDI-PORT.LerDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTLerCondition(80)};

fddiPORTUndesiredConnectionAttemptEvent **NOTIFICATION**
BEHAVIOUR
fddiPORTUndesiredConnectionAttemptEventBhv BEHAVIOUR
DEFINED AS "Generated when an undesired connection attempt has been made (see 5.2.4, 7.2.7, and 8.3).";;
WITH INFORMATION SYNTAX FDDI-PORT.UndesiredConnectionDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTUndesiredConnectionAttemptEvent(81)};

fddiPORTEBErrorCondition **NOTIFICATION**
BEHAVIOUR
fddiPORTEBErrorConditionBhv BEHAVIOUR
DEFINED AS "Generated when the Elasticity Buffer Error-Ct increments. This is handled as a condition in the Status Report Protocol. It is generated when an increment occurs in the station's sampling period (see 7.2.7 and 8.3).";;
WITH INFORMATION SYNTAX FDDI-PORT.EBErrorDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTEBErrorCondition(82)};

fddiPORTPathChangeEvent **NOTIFICATION**
BEHAVIOUR
fddiPORTPathChangeEventBhv BEHAVIOUR
DEFINED AS "This event is generated when the value of the fddiPORTCurrentPath changes. This event shall be surpressed if the value changes from isolated to local or local to isolated (see 7.2.7 and 8.3).";;
WITH INFORMATION SYNTAX FDDI-PORT.PathChangeDataType;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64) fddiPORTPathChangeEvent(83)};

fddiPORTVendorNotification **NOTIFICATION**
BEHAVIOUR
fddiPORTVendorNotificationBhv BEHAVIOUR
DEFINED AS "This notification is defined by the vendor. It provides the capability to define a vendor-specific notification based on the manufacturer's OUI. This action shall not duplicate or redefine any notification listed in the MIB.";;
WITH INFORMATION SYNTAX SET OF FDDI-COMMON.VendorMIBType;

```
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiPORT(64)
                    fddiPORTVendorNotification(252)};
```

6.4.5.7 ASN.1 definitions

6.4.5.7.1 Common definitions

```
FDDI-COMMON
    DEFINITIONS IMPLICIT TAGS::= BEGIN
    AvailablePaths                  ::= BIT STRING {
            primary                 (0),
            secondary               (1),
            local                   (2)} (SIZE (8))

    Counter::= INTEGER (0..4294967295)          -- 4 octets

            --  Represents a non-negative integer which monotonically increases until it
            --  reaches a maximum value, when it wraps around and starts increasing from
            --  zero.

    CurrentPath                     ::= ENUMERATED {
            isolated                (0),
            local                   (1),
            secondary               (2),
            primary                 (3),
            concatenated            (4),
            thru                    (5)}

    Flag                            ::= BOOLEAN   -- (True (1), False (0))

    LongAddressType                 ::= OCTET STRING (SIZE (6))

            --  SMT uses the MSB form for representation of 48-bit addresses (see
            --  4.1.2.3).

    PathConfig                      ::= SEQUENCE OF ResourceTag

    PermittedPaths                  ::= BIT STRING {
            local                   (0),
            secondary-alternate     (1),
            primary-alternate       (2),
            concatenated-alternate  (3),
            secondary-preferred     (4),
            primary-preferred       (5),
            concatenated-preferred  (6),
            thru                    (7)} (SIZE (8))

    PortType                        ::= ENUMERATED {
            a                       (0),
            b                       (1),
            s                       (2),
            m                       (3),
            none                    (4)}

    Present ::= ENUMERATED {
            notpresent              (0),
            present                 (1)}

    ResourceId                      ::= INTEGER (0..65535)      -- 2 octets

            --  An index used to refer to an instance of a MAC, PATH, or PORT, Resource
            --  ID; Indexing begins at 1. Zero is used to indicate all instances of an object.
```

```
ResourceTag                     ::= SEQUENCE {
        resource_type           ResourceType,
        resource_index          ResourceId,
        current_path            CurrentPath}

ResourceType                    ::= ENUMERATED{
        mac                     (2),
        port                    (4)}

SpecificData                    ::= SET {
        data                    ANY}

            --  The syntax for the data is defined by the vendor identified by the
            --  manufacturer OUI.

Time                            ::= INTEGER (0..4294967295)          -- 4 octets

            --  This data type specifies octet units of 80 nanoseconds as an integer value.
            --  It is used for timer and other attributes. The encoding is normal integer
            --  representation (not twos complement).

TimerTwosComplement             ::= OCTET STRING (SIZE (4))

            --  This data type is used for FDDI timers. They are encoded as unsigned
            --  two's complement integers, specifying time in octet units (80 nanoseconds)
            --  with the high order byte in the first octet of the string. This encoding is
            --  identical to the encoding of T_Bid in MAC Claim frames

TimeStamp                       ::= INTEGER (0..18446744073709551615)  -- 8
        octets

VendorMIBType                   ::= SEQUENCE {
        manufactureroui         OCTET STRING (SIZE(3)),

            --  SMT uses the MSB form for representation of Organizationally Unique
            --  Identifiers (see 4.1.2.3).

        vendoroctet             OCTET STRING (SIZE(1)),

            --  The vendor octet may be used to distinguish between different vendor
            --  specific attributes, actions and notifications.

        specificdata            SpecificData}
END -- End of FDDI COMMON syntax definitions
```

6.4.5.7.2 SMT definitions

```
FDDI-SMT
    DEFINITIONS IMPLICIT TAGS::= BEGIN
    ActionInfo                  ::= ENUMERATED {
        connect                 (0),
        disconnect              (1),
        path-Test               (2),
        self-Test               (3),
        disable-a               (4),
        disable-b               (5),
        disable-m               (6)}

    ActionInfoType              ::= SEQUENCE {
        actioninfo              ActionInfo}

    AvailablePathsType          ::= SEQUENCE {
        availablepaths          FDDI-COMMON.AvailablePaths}

    BypassPresentType           ::= SEQUENCE {
        bypasspresent           FDDI-COMMON.Flag}
```

```
Capabilities                    ::= BIT STRING {
        holdavailable           (0),
        cf-wrap-ab              (1)} (SIZE (16))

CF-State                        ::= ENUMERATED {
        isolated                (0),
        local_a                 (1),
        local_b                 (2),
        local_ab                (3),
        local_s                 (4),
        wrap_a                  (5),
        wrap_b                  (6),
        wrap_ab                 (7),
        wrap_s                  (8),
        c_wrap_a                (9),
        c_wrap_b                (10),
        c_wrap_s                (11),
        thru                    (12) }

CF-StateType                    ::= SEQUENCE {
        cf-state                CF-State}

ConfigCapabilitiesType          ::= SEQUENCE {
        capabilities            Capabilities}

ConfigPolicy                    ::= BIT STRING {
        configurationhold       (0),} (SIZE (16))

ConfigPolicyType                ::= SEQUENCE {
        configpolicy            Configpolicy}

Connection                      ::= BIT STRING {
        rejectA-A               (0),
        rejectA-B               (1),
        rejectA-S               (2),
        rejectA-M               (3),
        rejectB-A               (4),
        rejectB-B               (5),
        rejectB-S               (6),
        rejectB-M               (7),
        rejectS-A               (8),
        rejectS-B               (9),
        rejectS-S               (10),
        rejectS-M               (11),
        rejectM-A               (12),
        rejectM-B               (13),
        rejectM-S               (14),
        rejectM-M               (15)} (SIZE (16))

ConnectionPolicyType            ::= SEQUENCE {
        connection              Connection}

ConnectionPolicyConstraint      ::= ConnectionPolicyType (WITH COMPONENTS
                                {connection(32768..65535)})

ECMState                        ::= ENUMERATED {
        ec0                     (0),    -- Out
        ec1                     (1),    -- In
        ec2                     (2),    -- Trace
        ec3                     (3),    -- Leave
        ec4                     (4),    -- Path_Test
        ec5                     (5),    -- Insert
        ec6                     (6),    -- Check
        ec7                     (7)}    -- Deinsert
```

```
ECMStateType                    ::= SEQUENCE {
        ecmstate                ECMState}

HiVersionIdType                 ::= SEQUENCE {
        hiversionid             VersionType}

HiVersionIdConstraint           ::= HiVersionIdType (WITH COMPONENTS
                                {hiversionid(1..65535)})

HoldConditionDataType           ::= SEQUENCE {
        conditionstate          FDDI-COMMON.Flag,
        holdcondition           HoldConditionType}

HoldConditionType               ::= SEQUENCE {
        holdstate               HoldState}

HoldState                       ::= ENUMERATED {
        not-holding             (0),
        holding-prm             (1),    --holding on primary
        holding-sec             (2)}    --holding on secondary

HoldStateType                   ::= SEQUENCE {
        holdstate               HoldState}

LastSetStationIdType            ::= StationIdentifier

LoVersionIdType                 ::= SEQUENCE {
        loversionid             VersionType}

LoVersionIdConstraint           ::= LoVersionIdType (WITH COMPONENTS
                                {loversionid(1..65535)})

MAC-CtType                      ::= SEQUENCE {
        mac-Ct                  INTEGER (0..255)}               -- 1 octet

MACIndexesType                  ::= SET OF FDDI-COMMON.ResourceId

ManufacturerDataType            ::= SEQUENCE {
        manufacturerOUI         OCTET STRING (SIZE (3)),
        manufacturerData        OCTET STRING (SIZE (29))}

Master-CtType                   ::= SEQUENCE {
        master-ct               INTEGER (0..255)}               -- 1 octet

MIBVersionIdType                ::= SEQUENCE {
        mibversionid            VersionType}

NonMaster-CtType                ::= SEQUENCE {
        nonmaster-ct            INTEGER (0..255)}               -- 1 octet

NonMaster-CtConstraint          ::= NonMaster-CtType (WITH COMPONENTS
                                {nonmaster-ct(0..2)})

OpVersionIdType                 ::= SEQUENCE {
        opversionid             VersionType}

OpVersionIdConstraint           ::= OpVersionIdType (WITH COMPONENTS
                                {opversionid(1..65535)})

PeerWrapConditionDataType       ::= SEQUENCE {
        conditionstate          FDDI-COMMON.Flag,
        peerwrapcondition       PeerWrapConditionType}

PeerWrapConditionType           ::= SEQUENCE {
        cf-state                CF-State}

PeerWrapFlagType                ::= SEQUENCE{
        peerwrapflag            FDDI-COMMON.Flag}

PORTIndexesType                 ::= SET OF FDDI-COMMON.ResourceId
```

```
    RemoteDisconnectFlagType            ::= SEQUENCE {
        remotedisconnectflag            FDDI-COMMON.Flag}

    SetCountType                        ::= SEQUENCE {
        count                           FDDI-COMMON.Counter,
        settimestamp                    FDDI-COMMON.TimeStamp}

    StationIdentifier                   ::= SEQUENCE {
        implementeroctet                OCTET STRING (SIZE(2)),
        longaddress                     FDDI-COMMON.LongAddressType}

            --  The unique identifier of an FDDI station is composed of two octets that are
            --  implementer defined, followed by a six octet Universally Administered
            --  Address. The I/G bit and the U/L bit of the address shall be zero (see
            --  7.1.5).

    StationIdType                       ::= StationIdentifier

    Status                              ::= ENUMERATED {
        concatenated                    (0),
        separated                       (1),
        thru                            (2)}

    StationStatusType                   ::= SEQUENCE {
        status                          Status}

    StatRptPolicyType                   ::= SEQUENCE {
        flag                            FDDI-COMMON.Flag}

    T-NotifyType                        ::= SEQUENCE {
        t-notify                        INTEGER (0..65535)}          -- 2 octets

    T-NotifyConstraint                  ::= T-NotifyType (WITH COMPONENTS
                                        {t-notify(2..30)})

    TimeStampType                       ::= FDDI-COMMON.TimeStamp

    Trace-MaxExpirationType             ::= FDDI-COMMON.Time

    TransitionTimeStampType             ::= FDDI-COMMON.TimeStamp

    UserDataType                        ::= OCTET STRING (SIZE (32))

    VersionType                         ::= INTEGER (0..65535)
END -- End of FDDI SMT syntax definitions
```

6.4.5.7.3 MAC definitions

```
FDDI-MAC
    DEFINITIONS IMPLICIT TAGS       ::= BEGIN
    AvailablePathsType                  ::= SEQUENCE {
        availablepaths                  FDDI-COMMON.AvailablePaths}

    BridgeFunctionsType                 ::= SEQUENCE {
        bridgetype                      BridgeType}

    BridgeType                          ::= BIT STRING {
        tb                              (0),            -- Transparent Bridging Active
        sr                              (1),            -- Source Routing Active
        srt                             (2)} (SIZE (16))-- Source Routing Transparent
        Active

    Copied-CtType                       ::= FDDI-COMMON.Counter

    CurrentPathType                     ::= SEQUENCE {
        currentpath                     FDDI-COMMON.CurrentPath}
```

```
DA-FlagType                     ::= SEQUENCE {
        da-flag                 FDDI-COMMON.Flag}

DownstreamNbrType               ::= SEQUENCE {
        longaddress             FDDI-COMMON.LongAddressType}

DownstreamPORTType              ::= SEQUENCE {
        porttype                FDDI-COMMON.PortType}

DupAddressTest                  ::=     ENUMERATED {
        none                    (0),
        pass                    (1),
        fail                    (2)}

DupAddressTestType              ::= SEQUENCE {
        dupaddresstest          DupAddressTest}

DupCondition                    ::= BIT STRING{
        mydup                   (0),                    -- DA-Flag
        unadup                  (1)}  (SIZE(16))        -- UNDA-Flag

DuplicateAddressDataType        ::= SEQUENCE {
        conditionstate          FDDI-COMMON.Flag,
        duplicateaddresstype    DuplicateAddressType}

DuplicateAddressType            ::= SEQUENCE {
        dupcondition            DupCondition,
        fddilong                FDDI-COMMON.LongAddressType,
        fddiunalong             FDDI-COMMON.LongAddressType}

Error-CtType                    ::= FDDI-COMMON.Counter

Frame-CtType                    ::= FDDI-COMMON.Counter

FrameErrorFlagType              ::= SEQUENCE {
        frameerrorflag          FDDI-COMMON.Flag}

FrameErrorDataType              ::= SEQUENCE {
        conditionstate          FDDI-COMMON.Flag,
        frame-ct                FDDI-COMMON.Counter,
        error-ct                FDDI-COMMON.Counter,
        lost-ct                 FDDI-COMMON.Counter,
        frameerrorratio         FrameErrorRatioType}

FrameErrorRatioType             ::= SEQUENCE {
        frameerrorratio         INTEGER (0..65535)}             -- 2 octets

FrameErrorThresholdType         ::= SEQUENCE {
        frameerrorthreshold     INTEGER (0..65535)}             -- 2 octets

FS-Functions                    ::= BIT STRING {
        fs-repeating            (0),            -- MAC repeats C indicator as
                                                -- received, if  A_Flag is not set.
        fs-setting              (1),            -- MAC sets C indicator when
        copying
                                                -- with intent to forward, if A_Flag is
        not
                                                -- set and received A indicator is
        reset.
        fs-clearing             (2)} (SIZE (16))-- MAC sets A indicator and resets C
                                                -- indicator when frame was not
                                                -- copied, if A_Flag is set and
        received
                                                -- A  indicator is reset.

FrameStatusFunctionsType        ::= SEQUENCE {
        fsfunctions             FS-Functions}
```

```
HWPresentType                   ::= SEQUENCE {
        present                     FDDI-COMMON.Present}

Late-CtType                     ::= SEQUENCE {
        late-ct                     FDDI-COMMON.Counter}

LongAddressSetType              ::= SET OF FDDI-COMMON.LongAddressType

LongGrpAddressType              ::= LongAddressSetType

Lost-CtType                     ::= FDDI-COMMON.Counter

MAUnitdataAvailableType         ::= SEQUENCE {
        maunitdataavailable         FDDI-COMMON.Flag}

MAUnitdataEnableType            ::= SEQUENCE {
        maunitdataenable            FDDI-COMMON.Flag}

MACIndexType                    ::= SEQUENCE{
        macindex                    FDDI-COMMON.ResourceId}

NACondition                     ::= BIT STRING{
        unachange                   (0),
        dnachange                   (1)} (SIZE(8))

NeighbourChangeDataType         ::= SEQUENCE {
        multipleoccurrence          FDDI-COMMON.Flag,
        neighbourchangetype         NeighbourChangeType}

NeighbourChangeType             ::= SEQUENCE {
        nacondition                 NACondition,
        old-una                     FDDI-COMMON.LongAddressType,
        new-una                     FDDI-COMMON.LongAddressType,
        old-dna                     FDDI-COMMON.LongAddressType,
        new-dna                     FDDI-COMMON.LongAddressType,
        currentpath                 FDDI-COMMON.CurrentPath,
        smtaddress                  FDDI-COMMON.LongAddressType}

NotCopied-CtType                ::= FDDI-COMMON.Counter

NotCopiedFlagType               ::= SEQUENCE {
        notcopiedflag               FDDI-COMMON.Flag}

NotCopiedDataType               ::= SEQUENCE {
        conditionstate              FDDI-COMMON.Flag,
        notcopied-ct                FDDI-COMMON.Counter,
        copied-ct                   FDDI-COMMON.Counter,
        notcopiedratio              NotCopiedRatioType}

NotCopiedRatioType              ::= SEQUENCE {
        notcopiedratio              INTEGER (0..65535)}          -- 2 octets

NotCopiedThresholdType          ::= SEQUENCE {
        notcopiedthreshold          INTEGER (1..65535)}          -- 2 octets

OldDownstreamNbrType            ::= SEQUENCE {
        longaddress                 FDDI-COMMON.LongAddressType}

OldUpstreamNbrType              ::= SEQUENCE {
        longaddress                 FDDI-COMMON.LongAddressType}

PathChangeDataType              ::= SEQUENCE {
        multipleoccurrence          FDDI-COMMON.Flag,
        pathchangetype              PathChangeType}

PathChangeType                  ::= SEQUENCE {
        availablepaths              FDDI-COMMON.AvailablePaths,
        currentpath                 FDDI-COMMON.CurrentPath,
        requestedpaths              RequestedPathsType}
```

```
RequestedPathsType              ::= SEQUENCE {
        requestedpaths          FDDI-COMMON.PermittedPaths}

RingOp-CtType                   ::= SEQUENCE {
        ringop-ct               FDDI-COMMON.Counter}

RMTState                        ::=     ENUMERATED {
        rm0                     (0),    --Isolated
        rm1                     (1),    --Non_Op
        rm2                     (2),    --Ring_Op
        rm3                     (3),    --Detect
        rm4                     (4),    --Non_Op_Dup
        rm5                     (5),    --Ring_Op_Dup
        rm6                     (6),    --Directed
        rm7                     (7)}    --Trace

RMTStateType                    ::= SEQUENCE {
        rmtstate                RMTState}

ShortAddressSetType             ::= SET OF ShortAddressType

ShortAddressType                ::= OCTET STRING (SIZE (2))

ShortGrpAddressType             ::= SEQUENCE {
        shortgrpaddress         ShortAddressSetType}

SMTAddressType                  ::= SEQUENCE {
        smtaddress              FDDI-COMMON.LongAddressType}

T-MaxCapabilityType             ::= FDDI-COMMON.TimerTwosComplement

T-Pri0Type                      ::= FDDI-COMMON.TimerTwosComplement

T-Pri1Type                      ::= FDDI-COMMON.TimerTwosComplement

T-Pri2Type                      ::= FDDI-COMMON.TimerTwosComplement

T-Pri3Type                      ::= FDDI-COMMON.TimerTwosComplement

T-Pri4Type                      ::= FDDI-COMMON.TimerTwosComplement

T-Pri5Type                      ::= FDDI-COMMON.TimerTwosComplement

T-Pri6Type                      ::= FDDI-COMMON.TimerTwosComplement

TMaxType                        ::= FDDI-COMMON.TimerTwosComplement

TNegType                        ::= FDDI-COMMON.TimerTwosComplement

Token-CtType                    ::= FDDI-COMMON.Counter

Transmit-CtType                 ::= FDDI-COMMON.Counter

TReqType                        ::= FDDI-COMMON.TimerTwosComplement

TVXCapabilityType               ::= FDDI-COMMON.TimerTwosComplement

TvxExpired-CtType               ::= FDDI-COMMON.Counter

TvxValueType                    ::= FDDI-COMMON.TimerTwosComplement

UNDA-FlagType                   ::= SEQUENCE {
        unda-flag               FDDI-COMMON.Flag}

UpstreamNbrType                 ::= SEQUENCE {
        longaddress             FDDI-COMMON.LongAddressType}

END -- End of FDDI MAC syntax definitions
```

6.4.5.7.4 PATH definitions

```
FDDI-PATH
    DEFINITIONS IMPLICIT TAGS        ::= BEGIN
    ConfigurationType                ::= SEQUENCE {
        configuration                FDDI-COMMON.PathConfig}

    Index                            ::= INTEGER (0...65535){          -- 2 octets
        primary                      (1),
        secondary                    (2)}

            --    Local Paths are indexed using the values 3 to 255.

    IndexType                        ::= SEQUENCE {
        indextype                    Index}

    MaxT-ReqType                     ::= FDDI-COMMON.TimerTwosComplement

    RingLatencyType                  ::= FDDI-COMMON.Time

    SbaAvailableType                 ::= SEQUENCE {
        sbaavailable                 INTEGER (0..4294967295)}          -- 4 octets

    SbaAvailableConstraint           ::= SbaAvailableType (WITH COMPONENTS
                                     {sbaavailable (0..12500000)})

    SbaPayloadType                   ::= SEQUENCE {
        sbapayload                   INTEGER (0..4294967295)}          -- 4 octets

    SbaPayloadConstraint             ::= SbaPayloadType (WITH COMPONENTS
                                     {sbapayload (0..1562)})

    SbaOverheadType                  ::= INTEGER (0..4294967295)       -- 4 octets

    T-MaxLowerBoundType              ::= FDDI-COMMON.TimerTwosComplement

    T-RmodeType                      ::= FDDI-COMMON.Time

    TraceStatus                      ::= BIT STRING {
        traceinitiated               (0),
        tracepropagated              (1),
        traceterminated              (2),
        tracetimeout                 (3)} (SIZE(16))

    TraceStatusType                  ::= SEQUENCE {
        tracestatus                  TraceStatus}

    TVXLowerBoundType                ::= FDDI-COMMON.TimerTwosComplement
END -- End of FDDI PATH syntax definitions
```

6.4.5.7.5 PORT definitions

```
FDDI-PORT
    DEFINITIONS IMPLICIT TAGS        ::= BEGIN
    ActionInfoType                   ::= SEQUENCE {
        actiontype                   ActionType}

    ActionType                       ::= ENUMERATED {
        maintport                    (0),     --signal PC_Maint
        enableport                   (1),     --signal PC_Enable
        disableport                  (2),     --signal PC_Disable
        startport                    (3),     --signal PC_Start
        stopport                     (4)}     --signal PC_Stop
```

```
AvailablePathsType                ::= SEQUENCE {
        availablepaths                FDDI-COMMON.AvailablePaths}

BS-FlagType                       ::= SEQUENCE {
        bs-flag                       FDDI-COMMON.Flag}

ConnectionCapabilitiesType        ::=SEQUENCE {
        connectioncapabilities        ConnectionPolicies}

ConnectionPolicies                ::= BIT STRING {
        pc-mac-lct                    (0),
        pc-mac-loop                   (1)} (SIZE (8))

ConnectionPoliciesType            ::= SEQUENCE {
        connectionpolicies            ConnectionPolicies}

ConnectState                      ::= ENUMERATED {
        disabled                      (0),
        connecting                    (1),
        standby                       (2),
        active                        (3)}

ConnectStateType                  ::= SEQUENCE {
        connectstate                  ConnectState}

CurrentPathType                   ::= SEQUENCE {
        currentpath                   FDDI-COMMON.CurrentPath}

EBError-CtType                    ::= FDDI-COMMON.Counter

EBErrorDataType                   ::= SEQUENCE {
        conditionstate                FDDI-COMMON.Flag,
        eberrorcount                  FDDI-COMMON.Counter}

HWPresentType                     ::= SEQUENCE {
        present                       FDDI-COMMON.Present}

LCTFail-CtType                    ::= FDDI-COMMON.Counter

Lem-CtType                        ::= FDDI-COMMON.Counter

Lem-Reject-CtType                 ::= FDDI-COMMON.Counter

Ler-AlarmType                     ::= SEQUENCE {
        ler-alarm                     LerExponentType}

Ler-AlarmConstraint               ::= Ler-AlarmType (WITH COMPONENTS
                                      {ler-alarm(4..15)})

LerConditionType                  ::= SEQUENCE {
        lerflag                       FDDI-COMMON.Flag}

Ler-CutoffType                    ::= SEQUENCE {
        ler-cutoff                    LerExponentType}

Ler-CutoffConstraint              ::= Ler-CutoffType (WITH COMPONENTS
                                      {ler-cutoff(4..15)})

LerDataType                       ::= SEQUENCE {
        conditionstate                FDDI-COMMON.Flag,
        lertype                       LerType}

LerConstraint                     ::= LerType (WITH COMPONENTS {...,
                                      ler-cutoff(4..15), ler-alarm(4..15),
                                      ler-estimate(4..15)})

Ler-EstimateType                  ::= SEQUENCE {
        ler-estimate                  LerExponentType}

Ler-EstimateConstraint            ::= Ler-EstimateType (WITH COMPONENTS
                                      {ler-estimate(4..15)})
```

```
LerExponentType             ::= INTEGER (0..255)                -- 1 octet

LerFlagType                 ::= SEQUENCE{
    lerflag                     FDDI-COMMON.Flag}

LerType                     ::= SEQUENCE {
    ler-cutoff                  LerExponentType,
    ler-alarm                   LerExponentType,
    ler-estimate                LerExponentType,
    ler-reject-ct               FDDI-COMMON.Counter,
    ler-ct                      FDDI-COMMON.Counter}

LineState                   ::= ENUMERATED {
    qls                         (0),    --quiet
    ils                         (1),    --idle
    mls                         (2),    --master
    hls                         (3),    --halt
    pdr                         (4),    --receive = active, transmit = PDR
    lsu                         (5),    --receive = unknown
    nls                         (6)}    --receive = noise

MACIndicatedType            ::= SEQUENCE {
    t-val9                      FDDI-COMMON.Flag,
    r-val9                      FDDI-COMMON.Flag}

MACLoop-TimeType            ::= FDDI-COMMON.Time

MACLoop-TimeConstraint      ::= MACLoop-TimeType (WITH COMPONENTS
                                {macloop-time(2500000..4294967295)})

MACPlacementType            ::= SEQUENCE {
    macplacement                FDDI-COMMON.ResourceId}

Maint-LSType                ::= SEQUENCE {
    maint-LS                    LineState}

Maint-LSTypeConstraint      ::= Maint-LSType(WITH COMPONENTS
                                {maint-LS(0..4)})

MyTypeType                  ::= SEQUENCE {
    mytype                      FDDI-COMMON.PortType}

NeighbourTypeType           ::= SEQUENCE {
    neighbourtype               FDDI-COMMON.PortType}

PathChangeDataType          ::= SEQUENCE {
    multipleoccurrence          FDDI-COMMON.Flag,
    pathchangetype              PathChangeType}

PathChangeType              ::= SEQUENCE {
    availablepaths              FDDI-COMMON.AvailablePaths,
    currentpath                 FDDI-COMMON.CurrentPath,
    requestedpaths              RequestedPathsType,
    mytype                      FDDI-COMMON.PortType,
    neighbourtype               FDDI-COMMON.PortType}

PC-LSType                   ::= SEQUENCE {
    linestate                   LineState}

PCMState                    ::= ENUMERATED {
    pc0                         (0),    --Off
    pc1                         (1),    --Break
    pc2                         (2),    --Trace
    pc3                         (3),    --Connect
    pc4                         (4),    --Next
    pc5                         (5),    --Signal
    pc6                         (6),    --Join
```

```
            pc7                         (7),    --Verify
            pc8                         (8),    --Active
            pc9                         (9)}    --Maint

    PCMStateType                        ::= SEQUENCE {
            pcmstate                    PCMState}

    PC-Withhold                         ::= ENUMERATED {
            none                        (0),
            m-m                         (1),
            otherincompatible           (2),
            pathnotavailable            (3)}

    PC-WithholdType                     ::= SEQUENCE {
            pc-withhold                 PC-Withhold}

    PMDClass                            ::=     ENUMERATED {
            multimode                   (0),
            single-mode1                (1),
            single-mode2                (2),
            sonet                       (3),
            low-cost-fiber              (4),
            twisted-pair                (5),
            unknown                     (6),
            unspecified                 (7)}

    PMDClassType                        ::= SEQUENCE {
            pmdclass                    PMDClass}

    PortIndexType                       ::= SEQUENCE{
            portindex                   FDDI-COMMON.ResourceId}

    RequestedPathsType                  ::= SEQUENCE {
            none                        FDDI-COMMON.PermittedPaths,
            tree                        FDDI-COMMON.PermittedPaths,
            peer                        FDDI-COMMON.PermittedPaths}

    UndesiredConnectionDataType         ::= SEQUENCE {
            multipleoccurrence          FDDI-COMMON.Flag,
            undesiredconnectiontype     UndesiredConnectionType}

    UndesiredConnectionType             ::= SEQUENCE {
            porttype                    FDDI-COMMON.PortType,
            connectstate                ConnectState,
            pc-neighbour                FDDI-COMMON.PortType,
            pc-withhold                 PC-Withhold}
END -- End of FDDI PORT syntax definitions
```

6.4.5.8 Name binding

Figure 6 depicts the FDDI Naming Tree.

6.4.5.8.1 MAC naming

```
fddiMACName NAME BINDING
    SUBORDINATE OBJECT CLASS fddiMAC;
    NAMED BY SUPERIOR OBJECT CLASS fddiSMT;
    WITH ATTRIBUTE fddiMACIndex;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiNameBinding(6) fddiMAC(1)};
```

6.4.5.8.2 PATH naming

```
fddiPATHName NAME BINDING
    SUBORDINATE OBJECT CLASS fddiPATH;
    NAMED BY SUPERIOR OBJECT CLASS fddiSMT;
    WITH ATTRIBUTE fddiPATHIndex;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiNameBinding(6) fddiPATH(2)};
```

6.4.5.8.3 PORT naming

fddiPORTName NAME BINDING;
SUBORDINATE OBJECT CLASS fddiPORT;
NAMED BY SUPERIOR OBJECT CLASS fddiSMT;
WITH ATTRIBUTE fddiPORTIndex;
REGISTERED AS {iso(1) standard(0) iso9314(9314) fddiMIB(1) fddiNameBinding(6) fddiPORT(3)};

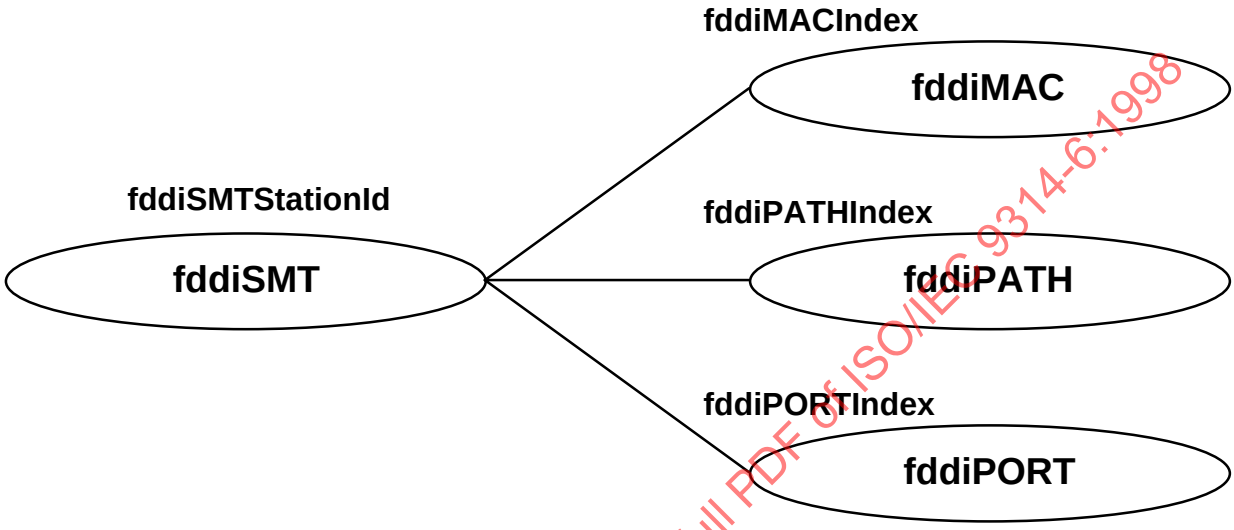
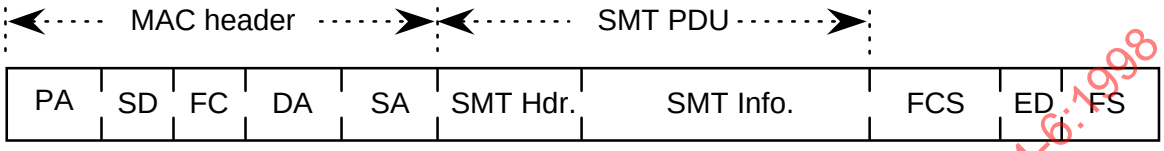


Figure 6 – FDDI naming tree

7 Facilities

This clause describes the frame formats that are used by SMT in the management of stations on an FDDI ring. A minimum of protocol information is included in this clause to put the frames into a context. The complete descriptions of the frame protocols that use these frames are in clause 8.

7.1 SMT frame format



Field	*Definition*	*Size*
PA	Preamble	4 or more symbols
SD	Starting delimiter	2 symbols
FC	Frame control	2 symbols
DA	Destination Address	12 symbols
SA	Source Address	12 symbols
SMT Hdr.	SMT protocol header	40 symbols
SMT Info.	SMT protocol information field	*N* symbols
FCS	Frame check sequence	8 symbols
ED	Ending delimiter	1 symbol
FS	Frame status	3 or more symbols

An SMT frame is one of the FDDI mechanisms that is used for peer–to–peer or *n*–layer management. The frame format has the three major components shown above. Within the MAC header, the Frame Control (FC) value identifies the frame as an SMT frame (FC=0100 0001 (SMT–Info) and FC=0100 1111 (Next Station Addressing (NSA))). The MAC header also carries the Destination and Source Address (DA and SA). The SMT Header identifies the frame's class and type and provides other information to supply the context for the SMT information in the frame. The Version ID serves as the tag for the format of the SMT Header.

The NSA FC value shall be used in neighbour information request and announcement frames (NIF request or NIF announcement) for the Neighbour Notification protocol (see 8.2). The NSA FC value may also be used in other request and announcement frames. The SMT–Info FC shall be used in all response frames, and may be used for all request and announcement frames, including the NIF frame when it is being used for purposes other than the Neighbour Notification protocol.

7.1.1 SMT frame contents

'Pad' fields occur in the SMT frames to provide natural alignment for each field and 32-bit alignment for SMT_Parameters. The pad octets shall be transmitted as zero(s). They shall be ignored by the receiving process.

7.1.2 SMT header

The SMT Header's format is shown below. Its fields are described in the text that follows the diagram.

SMT header field	Frame class	Frame type	Version ID	Transaction ID	Station ID	Pad	InfoField length
Length (octets)	1	1	2	4	8	2	2

7.1.2.1 Frame class and type

SMT frames are identified by their frame class and type The class identifies the function of the frame and the type designates whether the frame is an announcement, a request, or a response to a request. Table 1 summarizes the frame classes and the frame types that are defined for SMT.

7.1.2.2 Version ID

The SMT frame's Version ID field identifies the structure of the SMT Info field. To ensure the correct handling of SMT frames, the SMT frame header format shall be identical up to and including the Station ID field across all protocol versions. This allows all versions to deal with version identification and to respond correctly to version mismatches. The value of the Version ID for NIFs, SIFs, and ECFs shall be a constant value of X'00 01'. The value of the Version ID for all other SMT frames shall be the value of fddiSMTOpVersionId (see 6.4.5.4.1) and the frame shall be encoded according to the rules for that value of Version ID.

7.1.2.3 Transaction ID

The Transaction ID in the SMT header is used to pair SMT responses with their requests. The algorithm for generation of Transaction IDs for requests and announcements and the determination for their reuse are implementation dependent unless otherwise specified (see clause 8).

Table 1 – Summary of SMT frames

Frame_Class	Code	Frame_Types	Requirement
Neighbour Information Frames (NIF)	X'01'	Announcement Request Response	Optional Mandatory Mandatory
Status Information Frames (SIF)	-	-	-
SIF configuration	X'02'	Request Response	Optional Mandatory
SIF operation	X'03'	Request Response	Optional Mandatory
Echo Frames (ECF)	X'04'	Request Response	Optional Mandatory
Resource Allocation Frames (RAF)	X'05'	Request Response	Optional Optional
Request Denied Frame (RDF)	X'06'	Response	Mandatory
Status Report Frame (SRF)	X'07'	Announcement	Mandatory
Parameter Management Frames (PMF)	-	-	-
Get PMF	X'08'	Request Response	Optional Mandatory
Set PMF	X'09'	Request Response	Optional Optional
Extended Service Frames (ESF)	X'FF'	Announcement Request Response	Optional Optional Optional

7.1.2.4 Station ID

The Station ID is the unique identifier of the FDDI station transmitting an SMT frame. The Station ID is represented in a 64-bit field as X' yy yy xx xx xx xx xx xx'. The six least significant octets (xx) are a universally administered address in MSB representation. The I/G bit and the U/L bit of the address shall be zero. The two most significant octets (yy) are implementer defined.

7.1.2.5 InfoField length

The length of the SMT Information field is reported in the InfoField length. The value does not include the length of the SMT header or this field. Its value can be between zero and 4 458 octets.

MaxMACInfo = MaximumFrame - MACHeader - MinMACTrailer
= 4 500 - 16 - 6
= 4 478 octets

MaxSMTInfo = MaxMACInfo - SMTHeader
= 4 478 - 20
= 4 458 octets

SMT shall be capable of receiving and processing frames of the maximum size supported by ISO/IEC 9314-2 on MAC and ISO/IEC 9314-1 on PHY (4 478 octets).

The length of the MAC INFO field of frames transmitted by SMT shall not exceed 4 352 bytes, excluding ECHO request and ECHO response frames, which may be transmitted with MAC INFO length of the maximum size supported by ISO/IEC 9314-2 on MAC and ISO/IEC 9314-1 on PHY.

The maximum MAC INFO field for SMT PDUs of 4 352 octets was chosen for consistency with other standards. It is important to note that the maximum length of some SMT_Parameters could exceed this value. Furthermore, combinations of SMT_Parameters transmitted in a single SMT PDU could also exceed this maximum.

7.1.3 SMT InfoField

The SMT information field format is defined by the following diagram and text.

	$SMT\_Parameter_1$				$SMT\_Parameter_2$				
SMT Info Field	Parameter Type	Parameter Length	Resource Index	Parameter Value	Parameter Type	Parameter Length	Resource Index	Parameter Value	…
Length (octets)	2	2	4	N_1	2	2	4	N_2	…

Parameter_Type A 2-octet value that identifies the SMT_Parameter or attribute. An SMT_Parameter marked OPTIONAL means that it may be included in the frame or may be totally absent.

MIB attributes (see 6.4) have Parameter_Types of the form 'xx zz' (where xx is the FDDI MIB object identifier and zz is the attribute identifier). The encoding of the MIB attributes is specified in the SMT encoding rules (see 7.1.4).

SMT_Parameters not included in the MIB are also used in SMT frames and have Parameter_Types of the form '00 zz'. The encoding of these parameters is specified in the Parameter_Types (see 7.3).

Parameter_Length A 2-octet binary number that gives the length, in octets, of the Parameter_Value and Resource_Index fields. The length does not include the length of the Parameter_Type or the Parameter_Length fields.

Resource_Index A 4-octet integer value that identifies the object instance. This field shall be included when referencing MIB attributes associated with the MAC, Path, and Port object classes. This field shall be omitted for '00 zz' Parameter_Types and MIB attributes of the SMT object class.

Parameter_Value A unit of SMT information. This field shall not be included in PMF Get Request Frames, except when the Get is for a vendor specific attribute, but is required for all other SMT frame classes and types. The encoding of the value is defined by the SMT encoding rules.

The ordering of SMT_Parameters in the SMT InfoField is not specified and they may occur in any order.

7.1.4 SMT encoding rules

The way in which management information is conveyed between systems is through the use of a transfer syntax which describes the rules by which the data will be formatted for communication. There are a number of possible transfer syntax rules. This International Standard describes the SMT Encoding Rules. The application of this transfer syntax to the ASN.1 representation found in the SMT MIB will yield the format for communication. This transfer syntax differs from the syntax defined in ISO 8825, hereafter referred to as BER.

The application of the SMT Encoding Rules to the ASN.1 (see 6.4.5.7) yields the proper encoding of the Parameter_Value subfield of the SMT InfoField (see 7.1.3). The encoding of the ASN.1 is dependent on the data type described in the ASN.1 definition. In the case of complex data types, the underlying data type is encoded first, then the containing data type is encoded.

The following encoding rules shall be applied to the data types indicated in this part of ISO/IEC 9314.

INTEGER An INTEGER shall consist of *n* octets, where *n* is the minimum number of octets required to accommodate the range indicated in the value constraint stated in the ASN.1 definition. The octets shall be an unsigned binary number equal to the integer and shall be transmitted most-significant octet first.

ENUMERATED An ENUMERATED shall consist of 2 octets. The octets shall be encoded as an unsigned binary number equal to the value indicated and shall be transmitted most-significant octet first.

BIT STRING A BIT STRING shall consist of *n*/8 octets, where *n* is specified by the SIZE constraint of the ASN.1 definition. The bit-string value is encoded by setting the appropriate bit(s) in the encoding where the first bit of the bit-string value is encoded into the first bit of the encoding. The bits within a BIT STRING are numbered from 0 to *n*-1, where bit *n*-1 of the octet is the first bit transmitted.

OCTET STRING An OCTET STRING shall consist of *n* octets, where *n* is specified by the SIZE constraint of the ASN.1 definition. The octet string value shall be encoded commencing with the most-significant octet and proceeding to the trailing octet.

BOOLEAN A BOOLEAN shall consist of one octet. A false value is encoded as zero and a true value is encoded as one.

ANY An ANY encoding is not restricted by these encoding rules.

SEQUENCE A SEQUENCE shall consist of *n* octets, where *n* is the number of octets required to encode the listed data types. The value for *n* shall be a multiple of 4. A padding of octets containing a zero value shall be added preceding the first value encoded to insure that n is a multiple of 4. A SEQUENCE encoding shall consist of one data value from each of the types listed in the ASN.1 definition in order of appearance.

SEQUENCE OF A SEQUENCE OF shall consist of *n* octets, where *n* is the number of octets required to encode the list of values. The value for n shall be a multiple of 4. A padding of octets containing a zero value shall be added following the last value encoded to insure that *n* is a multiple of 4. A SEQUENCE OF

value shall consist of zero, one or more complete encodings of data values from the type listed in the ASN.1. The order of data values shall be preserved.

SET OF — A SET OF shall consist of *n* octets, where *n* is the number of octets required to encode the list of values. The value for *n* shall be a multiple of 4. A padding of octets containing a zero value shall be added following the last value encoded to insure that *n* is a multiple of 4. A SET OF value shall consist of zero, one or more complete encodings of data values from the type listed in the ASN.1. The order of data values need not be preserved.

The following is an example application of the SMT Encoding Rules.

The application of the SMT Encoding Rules to the fddiMACFrameErrorCondition notification is described in the following text. This example encoding is supplied here simply as an example of the process involved in creating the encoding for an attribute, action, or notification.

The GDMO specification for fddiMACFrameErrorCondition is found in the MAC Notification Templates (see 6.4.5.6.2). Note that in this subclause, as in all template subclausess, items are listed in order of their registration number (e.g. fddiMAC 141). The "WITH INFORMATION SYNTAX" clause in the fddiMACFrameErrorCondition template indicates the ASN.1 definition of this notification. In this case the referenced definition is FDDI-MAC.FrameErrorDataType located in the MAC ASN.1 definitions (see 6.4.5.7.3). Note that in this subclause, as in all ASN.1 definition subclauses, items are listed in alphabetical order.

FrameErrorDataType is a SEQUENCE with five elements. The encoding process begins with the last element, which in this case is frameerrorratio. This element is defined to be the FrameErrorRatioType data type. FrameErrorRatioType, also a MAC definition, is a SEQUENCE consisting of one element, frameerrorratio. This element is defined to be an INTEGER whose allowable range is from 0 to 65535, which can be represented by two octets. Since this is a SEQUENCE, the two octets representing the INTEGER must be preceded by a two-octet pad of zeros to insure a length that is a multiple of 4.

The next three items from the bottom of FrameErrorDataType are all of type FDDI-COMMON.Counter, which is shown in the Common Definitions (see 6.4.5.7.1). Counter is an INTEGER from 0 to 4294967295, which can be represented by 4 octets. Thus, each of these fields are 4 octets in length.

The last item to encode is the first in the list and is conditionstate, which is of type FDDI-COMMON.Flag. Flag is of type BOOLEAN, which is described (see 7.1.4) as being one octet, with a value of either zero or one. This ends the encoding of the items in the FrameErrorDataType SEQUENCE, but since the number of octets is 17 (not a multiple of 4), a three-octet pad of zeros must be added at the beginning of the SEQUENCE.

Since this is a MAC notification, the parameter encoding must be preceded by a MAC Index, which is a 4 octet integer value (see 7.1.3). The parameter length can now be calculated by adding the MAC Index field length plus the length of the FrameErrorDataType structure. For this example, the value is 24, which maps to a 2 octet parameter length field of X'0018'. The parameter type value, in this case X'208D', can then be obtained from the MIB Summary (see 6.4.5.1).

Thus, for this example, the final encoding for the parameter, fddiMACFrameErrorCondition, is as follows:

Parameter Type	X'20 8D'
Parameter Length	X'00 18'
macindex	X'00 00 00 01'-X'00 00 FF FF'
Pad	X'00 00 00'
conditionstate	X'00'-X'01'
frame-ct	X'xx xx xx xx'
error-ct	X'xx xx xx xx'
lost-ct	X'xx xx xx xx'
Pad	X'00 00'
frameerrorratio	X'xx xx'

7.1.5 Byte ordering in multibyte fields

The octets of multibyte fields are ordered in the frames most significant byte transmitted first with the most significant bit of that byte transmitted first.

For example, the decimal number 262 (X'106') is ordered as the following (where Octet 1 is closest to the beginning of the frame).

Octet 1	Octet 2
0000 0001	0000 0110

7.1.6 Addressing

With the exception of specifying an individual address as the required address in a response frame, there is no restriction on addressing modes (broadcast, group, multicast) used for SMT frames. Similarly, except for the specification of the NSA FC for the Neighbour Notification protocol and the SMT_Info FC for all response frames, there is no restriction on the use of FC values allowed (NSA and SMT_Info). This was done to support potentially useful, application specific modes of collecting management information.

NOTE 8 The use of group or broadcast addresses in the destination address of SMT frames (other than frames sent with Next Station Addressing (FC=0100 1111)) is likely to generate substantial numbers of responses. While it is permitted to send such frames, implementers are urged to exercise caution and analyze carefully the possible adverse consequences.

Long addresses (48 bits) are used in the address fields in the SMT frames.

SMT defines the use of several IEEE assigned multicast addresses. SMT also uses a universally administered address to indicate that the value of an address attribute is not known. These addresses are identified by a name. Throughout this International Standard all SMT defined addresses will be referenced by their names. Their assigned values are shown in the following table in both Canonical and Most Significant Bit first (MSB) representations. Observe that for canonical representation of addresses, hyphens are used between octets. For MSB representation of addresses, colons are used between octets.

Name	Canonical Representation	MSB Representation
SMT-Directed-Beacon-DA	01-80-C2-00-01-00	80:01:43:00:80:00
SMT-SRF-DA	01-80-C2-00-01-10	80:01:43:00:80:08
All FDDI Concentrator MACs	01-80-C2-00-01-20	80:01:43:00:80:04
SMT-SBA-DA	01-80-C2-00-01-30	80:01:43:00:80:0C
SMT-Unknown-Address	00-00-F8-00-00-00	00:00:1F:00:00:00

Although assigned, the address "All FDDI Concentrators MACs" has no use specified in this part of ISO/IEC 9314. It is included in this list of addresses for information purposes only.

7.1.7 Frame validity

The SMT frame protocols shall only operate on SMT frames (FC = X'41' or '4F') that meet MAC validity rules, have been received with good FCS, have the E indicator received as R (reset), and have the A and C indicators received as R (reset) or S (set). In addition, SMT frames that have been transmitted using next station addressing (FC = X '4F') shall only be acted on by SMT protocols if the A indicator is received as R (reset). These requirements are not intended to prevent a station from copying any frame for other purposes.

7.2 SMT frames

This subclause describes the formats of SMT frames used by the SMT protocols (see clause 8). Note that the SMT_Parameters shown in the frame definitions (see 7.3) may be included in any order within the SMT Information Field. The order as presented in the frame definitions holds no particular significance. The order of the fields in the SMT header portion of the frame is fixed.

7.2.1 Neighbour Information Frame (NIF)

The Neighbour Information Frame (NIF) is used by a station for periodic announcement of its address and basic station description.

NIFs are used in the Neighbour Notification protocol (see 8.2). The Neighbour Notification protocol allows a MAC to determine its logical upstream neighbour address (UNA) and its logical downstream neighbour address (DNA). The protocol also detects a duplicated fddiMACSMTAddress on an operational ring.

NIFs may be used by a monitoring station to build a logical ring map. The monitoring station can do this by monitoring the periodic NIF announcements or requests (broadcasts) or by using the request/response NIF facility directly (see 8.2.1).

7.2.1.1 NIF announcement OPTIONAL

The NIF announcement frame values are as follows:

FC:	X'4F'	(NSA)
DA:	X'FF FF FF FF FF FF'	(Broadcast)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'01'	(NIF)
Frame_Type:	X'01'	(Announcement)
Version_ID:	X'00 01'	(Constant)
Transaction_ID:	X'xx ... xx' (4 octets)	(Source station defined)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length	X'00 28'	
NIF information field:		
SMT_Parameter	X'00 01'	(Upstream Neighbour Address)
SMT_Parameter	X'00 02'	(Station descriptor)
SMT_Parameter	X'00 03'	(Station state)
SMT_Parameter	X'20 0B'	(fddiMACFrameStatusFunctions)

7.2.1.2 NIF request

The NIF request frame values are as follows:

FC:	X'4F'	(NSA)
	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Broadcast, Group, or Individual)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'01'	(NIF)
Frame_Type:	X'02'	(Request)
Version_ID:	X'00 01'	(Constant)
Transaction_ID:	X'xx ... xx' (4 octets)	(Source station defined)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	X'00 28'	
NIF information field:		
SMT_Parameter	X'00 01'	(Upstream Neighbour Address)
SMT_Parameter	X'00 02'	(Station descriptor)
SMT_Parameter	X'00 03'	(Station state)
SMT_Parameter	X'20 0B'	(fddiMACFrameStatusFunctions)

7.2.1.3 NIF response

The NIF response frame values are as follows:

FC:	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Individual from request SA)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'01'	(NIF)
Frame_Type:	X'03'	(Response)
Version_ID:	X'00 01'	(Constant)
Transaction_ID:	X'xx ... xx' (4 octets)	(Transaction_ID from request frame)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	

InfoField_Length:	X'00 28'	
NIF information field:		
SMT_Parameter	X'00 01'	(Upstream Neighbour Address)
SMT_Parameter	X'00 02'	(Station descriptor)
SMT_Parameter	X'00 03'	(Station state)
SMT_Parameter	X'20 0B'	(fddiMACFrameStatusFunctions)

7.2.2 Status Information Frames (SIF)

Status Information Frames (SIF) are used, in conjunction with the SIF protocol (see 8.5), to request and provide, in response, a station's configuration and operating information. There are two classes of SIFs to provide this function; the SIF Configuration and the SIF operation request and response frames. Potential uses for these frames include creating a physical ring map, fault isolation, and statistics monitoring.

7.2.2.1 SIF configuration request OPTIONAL

FC:	X'4F'	(NSA)
	or X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Broadcast, Group, or Individual)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'02'	(SIF configuration)
Frame_Type:	X'02'	(Request)
Version_ID:	X'00 01'	(Constant)
Transaction_ID:	X'xx ... xx' (4 octets)	(Source station defined)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	X'00 00'	
SIF Information Field:		
	NULL	

7.2.2.2 SIF configuration response

FC:	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Individual from request SA)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'02'	(SIF configuration)
Frame_Type:	X'03'	(Response)
Version_ID:	X'00 01'	(Constant)
Transaction_ID:	X'xx ... xx' (4 octets)	(Transaction_ID from request frame)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)
SIF information field:		
SMT_Parameter	X'00 04'	(MsgTimeStamp)
SMT_Parameter	X'00 02'	(Station descriptor)
SMT_Parameter	X'00 14'	(SMT supported versions)
SMT_Parameter	X'00 03'	(Station state)
SMT_Parameter	X'00 05'	(Station policies)
SMT_Parameter	X'00 06'	(Path latency.contribution/per ring – OPTIONAL)
SMT_Parameter	X'00 07'	(MAC neighbours – One entry per MAC physically present in the station)
SMT_Parameter	X'00 08'	(Path descriptor)
SMT_Parameter	X'10 35'	(Set Count – OPTIONAL, but mandatory if parameter management set is supported, ref. fddiSMTSetCount)

7.2.2.3 SIF operation request OPTIONAL

FC:	X'4F'	(NSA)
	or X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Broadcast, Group, or Individual)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'03'	(SIF operation)
Frame_Type:	X'02'	(Request)
Version_ID:	X'00 01'	(Constant)
Transaction_ID:	X'xx ... xx' (4 octets)	(Source station defined)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	X'00 00'	
SIF information field:		
	NULL	

7.2.2.4 SIF operation response

FC:	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Individual from request SA)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'03'	(SIF operation)
Frame_Type:	X'03'	(Response)
Version_ID:	X'00 01'	(Constant)
Transaction_ID:	X'xx ... xx' (4 octets)	(Transaction_ID from request frame)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)
SIF Information Field:		
SMT_Parameter	X'00 04'	(MsgTimeStamp)
SMT_Parameter	X'00 09'	(MAC status – One parameter entry per MAC physically present in station)
SMT_Parameter	X'00 0A'	(Port LEM status – one parameter entry per Port physically present in station)
SMT_Parameter	X'00 0B'	(MAC frame count – one parameter entry per MAC physically present in station)
SMT_Parameter	X'00 0C'	(MAC frame not copied count – OPTIONAL one parameter entry per MAC physically present in station)
SMT_Parameter	X'00 0D'	(MAC priority values – OPTIONAL one parameter entry per MAC physically present in station)
SMT_Parameter	X'00 0E'	(Port EB status – OPTIONAL one parameter entry per Port physically present in station)
SMT_Parameter	X'00 0F'	(Manufacturer field – OPTIONAL)
SMT_Parameter	X'00 10'	(User Field)
SMT_Parameter	X'10 35'	(Set Count – OPTIONAL, but mandatory if parameter management set is supported, ref. fddiSMTSetCount)

7.2.3 ECHO Frame (ECF)

The SMT Echo frames (ECF) are defined for SMT-to-SMT loopback testing on an FDDI ring using the ECHO protocol (see 8.6). The SMT ECHO frames may contain any amount up to the maximum frame size supported by ISO/IEC 9314 (see 7.1.2.5) of implementation-specific Echo Data. Potential uses for these frames include confirming that a station's Port, MAC, and SMT are operational, and testing for data–sensitive network failures by placing suspect data patterns in the ECHO data field.

7.2.3.1 Echo request frame OPTIONAL

FC:	X'4F'	(NSA)
	or X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Broadcast, Group, or Individual)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'04'	(ECF)
Frame_Type:	X'02'	(Request)
Version_ID:	X'00 01'	(Constant)
Transaction_ID:	X'xx ... xx' (4 octets)	(Source station defined)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)
ECF information field:		
SMT_Parameter	X'00 11'	(Echo Data)

7.2.3.2 Echo response frame

FC:	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Individual from request SA)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'04'	(ECF)
Frame_Type:	X'03'	(Response)
Version_ID:	X'00 01'	(Constant)
Transaction_ID:	X'xx ... xx' (4 octets)	(Transaction_ID from request frame)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)
ECF Information Field:		
SMT_Parameter	X'00 11'	(Echo Data)

7.2.4 Resource Allocation Frame (RAF) OPTIONAL

Resource Allocation Frames are defined to support a variety of network policies for allocation of resources. In this specification, the only identified resource is synchronous bandwidth (see 8.7).

7.2.4.1 SBA resource allocation request frame OPTIONAL

FC:	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Broadcast, Group, or Individual)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'05'	(RAF)
Frame_Type:	X'02'	(Request)
Version_ID:	X'00 02'	(This International Standard)
Transaction_ID:	X'xx ... xx' (4 octets)	(Source station defined)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)

RAF information field:

SMT_Parameter	X'00 15'	(Resource type – indicates synchronous bandwidth)
SMT_Parameter	X'00 16'	(SBA command)

SBA command value of X'00 00 00 01' (Request Allocation)

SMT_Parameter	X'32 0B'	(Path Type – Path requesting allocation, fddiPATHIndex)
SMT_Parameter	X'00 17'	(SBA payload request – signed number specifying the amount of allocation or deallocation)
SMT_Parameter	X'00 18'	(SBA overhead request – signed number specifying the amount of allocation or deallocation)
SMT_Parameter	X'32 0F'	(Current SBA payload for this Path, fddiPATHSbaPayload)

SMT_Parameter	X'32 10'	(Current SBA overhead for this Path, fddiPATHSbaOverhead)
SMT_Parameter	X'00 19'	(Allocation address - individual or group address responsible for the allocation, default = SA of the request)
SMT_Parameter	X'00 1A'	(Category - of the allocation)
SMT_Parameter	X'00 1B'	(Max T_Neg - longest T_Neg acceptable for the request)
SMT_Parameter	X'00 1C'	(Min segment - minimum segment size)

SBA command value of X'00 00 00 02' *(Report Allocation)*

No additional parameters.

SBA command value of X'00 00 00 03' *(Change Allocation)*

SMT_Parameter	X'32 0B'	(Path Type – Path on which change is requested, fddiPATHIndex)
SMT_Parameter	X'32 0F'	(New current SBA payload for this Path, fddiPATHSbaPayload)
SMT_Parameter	X'32 10'	(New current SBA overhead for this Path, fddiPATHSbaOverhead)
SMT_Parameter	X'00 1A'	(Category - of the allocation for the change. A value of zero is not category specific.)

7.2.4.2 SBA resource allocation response frame OPTIONAL

FC:	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Individual from request SA)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'05'	(RAF)
Frame_Type:	X'03'	(Response)
Version_ID:	X'00 02'	(This International Standard)
Transaction_ID:	X'xx ... xx' (4 octets)	(From request)
Station_ID:	X'yy yy xx xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)

RAF information field:

SMT_Parameter	X'00 15'	(Resource type – indicates synchronous bandwidth)
SMT_Parameter	X'00 16'	(SBA command – from request SBA command)

SBA command value of X'00 00 00 01' *(Request Allocation)*

SMT_Parameter	X'00 12'	(Reason code)
SMT_Parameter	X'32 0B'	(Path type – Path requesting allocation, fddiPATHIndex)
SMT_Parameter	X'32 0F'	(Current SBA payload for this Path, fddiPATHSbaPayload,)
SMT_Parameter	X'32 10'	(Current SBA overhead for this Path, fddiPATHSbaOverhead)
SMT_Parameter	X'00 19'	(Allocation address - from the request)
SMT_Parameter	X'00 1A'	(Category - from the request)
SMT_Parameter	X'00 1D'	(SBA allocatable – maximum amount management process has left to allocate)

SBA command value of X'00 00 00 02' *(Report Allocation)*

SMT_Parameter	X'32 0B'	(Path type – Primary Path, fddiPATHIndex)
SMT_Parameter	X'32 0F'	(Current SBA payload for the Primary Path, fddiPATHSbaPayload)
SMT_Parameter	X'32 10'	(Current SBA overhead for the Primary Path, fddiPATHSbaOverhead)
SMT_Parameter	X'32 0B'	(Path type – OPTIONAL for the Secondary Path, fddiPATHIndex)

SMT_Parameter	X'32 0F'	(Current SBA payload– OPTIONAL for the Secondary Path, fddiPATHSbaPayload)
SMT_Parameter	X'32 10'	(Current SBA overhead – OPTIONAL for the Secondary Path, fddiPATHSbaOverhead)

SBA command value of X'00 00 00 03' *(Change Allocation)*

SMT_Parameter	X'32 0B'	(Path type – from the request, fddiPATHIndex)
SMT_Parameter	X'32 0F'	(Current SBA payload – after the change, fddiPATHSbaPayload)
SMT_Parameter	X'32 10'	(Current SBA overhead – after the change, fddiPATHSbaOverhead)
SMT_Parameter	X'00 1A'	(Category - from the request)

7.2.5 Request Denied Frame (RDF)

The Request Denied Frame is defined for notification of SMT request frame format and protocol errors (see clause 8).

FC:	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Individual from request SA)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'06'	(RDF)
Frame_Type:	X'03'	(Response)
Version_ID:	X'00 02'	(This International Standard)
Transaction_ID:	X'xx ... xx' (4 octets)	(Transaction_ID from request frame)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)

RDF information field:

SMT_Parameter	X'00 12'	(Reason Code)
SMT_Parameter	X'00 14'	(SMT Supported Versions – OPTIONAL but mandatory when reason code is 'frame version not supported')
SMT_Parameter	X'00 13'	(Rejected frame beginning – OPTIONAL but mandatory when reason code is 'frame version not supported')

7.2.6 Extended Service Frame (ESF) OPTIONAL

The Extended Service Frame (ESF) is defined for extending and exercising new SMT services. The protocol for the ESF is defined by the owner of the ESF_ID (see 8.8).

FC:	X'4F'	(NSA)
	or X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Broadcast, Group, or Individual)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'FF'	(ESF)
Frame_Type:	X'01'	(Announcement)
	or X'02'	(Request)
	or X'03'	(Response)
Version_ID:	X'00 02'	(This International Standard)
Transaction_ID:	X'xx ... xx' (4 octets)	(Announcement: Source station defined Request: Source station defined Response: Transaction_ID from request frame)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)

ESF Information Field:

SMT_Parameter	X'FF FF'	(ESF_ID, the ESF Identification Code – Shall be first in the info field.)
SMT_Parameter	X'xx xx'	(Any number of other parameters. The meaning of these parameters depends on the ESF_ID value.)

7.2.7 Status Report Frame (SRF)

The Status Report Frame is used by a station to announce Station Status which may be of interest to a manager of an FDDI ring using the Status Report protocol (see 8.3).

FC:	X'41'	(SMT Info)
DA:	SMT-SRF-DA	(SRF multicast address)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'07'	(Status report)
Frame_Type:	X'01'	(Announcement)
Version_ID:	X'00 02'	(This International Standard)
Transaction_ID:	X'xx ... xx' (4 octets)	(Source station defined)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)
SRF Information Field:		
SMT_Parameter	X'10 33'	(fddiSMTTimeStamp)
SMT_Parameter	X'10 34'	(TransitionTimeStamp; fddiSMTTransitionTimeStamp)
SMT_Parameter	X'xx xx'	(Event or condition as specified in SRF protocol (see 8.3). There may be multiple parameters corresponding to events and conditions.)

7.2.8 Parameter Management Frames (PMF)

Parameter Management Frames provide the means for remote access to station attributes via the Parameter Management protocol (see 8.4). The Parameter Management protocol operates on all SMT Management Information Base (MIB) attributes.

7.2.8.1 PMF Get request frame OPTIONAL

FC:	X'4F'	(NSA)
	or X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Broadcast, Group, or Individual)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'08'	(PMF Get)
Frame_Type:	X'02'	(Request)
Version_ID:	X'00 02'	(This International Standard)
Transaction_ID:	X'xx ... xx' (4 octets)	(Source station defined)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)
PMF information field:		
SMT_Parameter	X'xx xx'	(Including the Parameter_Type, Parameter_Length and Resource_Index, if appropriate, of the requested parameter. The Parameter_Value field may be omitted (see 7.1.3). There may be multiple SMT_Parameters.)

7.2.8.2 PMF Get response frame

FC:	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Individual from request SA)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'08'	(PMF Get)
Frame_Type:	X'03'	(Response)
Version_ID:	X'00 02'	(This International Standard)
Transaction_ID:	X'xx ... xx' (4 octets)	(Transaction_ID from request frame)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)
PMF information field:		
SMT_Parameter	X'00 12'	(Reason code)
SMT_Parameter	X'10 33'	(fddiSMTTimeStamp)
SMT_Parameter	X'10 35'	(SetCount current value (fddiSMTSetCount). OPTIONAL Included if PMF Set protocol is implemented.)
SMT_Parameter	X'10 36'	(LastSetStationID (fddiSMTLastSetStationID). OPTIONAL Included if PMF Set protocol is implemented.)
SMT_Parameter	X'xx xx'	(Requested SMT_Parameter with the current value (see 7.1.3). There may be multiple SMT_Parameters.

Note that some or all of the requested SMT_Parameter(s) may be omitted if the reason code is not Success.

7.2.8.3 PMF Set request frame OPTIONAL

FC:	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Individual)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'09'	(PMF Set Frame)
Frame_Type:	X'02'	(Request)
Version_ID:	X'00 02'	(This International Standard)
Transaction_ID:	X'xx ... xx' (4 octets)	(Source station defined)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)
PMF information field:		
SMT_Parameter	X'00 21'	(Authorization OPTIONAL)
SMT_Parameter	X'10 35'	(SetCount OPTIONAL, see 8.4.1.1)
SMT_Parameter	X'xx xx'	(The target attribute, including the desired value. One permitted per PMF Set request.)

7.2.8.4 PMF Set response frame OPTIONAL

FC:	X'41'	(SMT Info)
DA:	X'xx ... xx' (6 octets)	(Individual from request SA)
SA:	X'xx ... xx' (6 octets)	(Individual)
Frame_Class:	X'09'	(PMF Set Frame)
Frame_Type:	X'03'	(Response)
Version_ID:	X'00 02'	(This International Standard)
Transaction_ID:	X'xx ... xx' (4 octets)	(From request frame)
Station_ID:	X'yy yy xx xx xx xx xx'	(see 7.1.2.4)
Pad:	X'00 00'	
InfoField_Length:	Variable	(see 7.1.2.5)
PMF Information Field:		
SMT_Parameter	X'00 12'	(Reason code)
SMT_Parameter	X'10 33'	(fddiSMTTimeStamp)

SMT_Parameter	X'10 35'	(New SetCount, fddiSMTSetCount)
SMT_Parameter	X'10 36'	(LastSetStationID, - fddiSMTLastSetStationID)
SMT_Parameter	X'xx xx'	(The new value of the target attribute)

Note that the last SMT_Parameter may optionally be omitted for reason codes other than Success and BadSetCount, and for Actions (see 6.4.5.5).

7.3 SMT_Parameters

The SMT_Parameters with Parameter_Type encodings X'00 xx' specified in the SMT frames (see 7.2) are described here in more detail. SMT_Parameters with encodings of X'yy xx' where 'yy' is not '00' are described in the MIB (see 6.4). Some of the X'00 xx' parameters are derived from MIB attributes. For these, a reference to the MIB attribute is given. Other X'00 xx' parameters do not map to any MIB attributes. The encoding and descriptions are given in this subclause.

7.3.1 General parameters

7.3.1.1 Upstream Neighbour Address (UNA)

Parameter_Type	X'00 01'	
Parameter_Length	X'00 08'	
Parameter_Value		
Pad	X'00 00'	
UNA	X'xx ... xx' (6 octets)	(Individual address)
Reference	UNA: fddiMACUpstreamNbr	

7.3.1.2 Station descriptor

Parameter_Type	X'00 02'	
Parameter_Length	X'00 04'	
Parameter_Value		
NodeClass	X'00'	(Station = an FDDI node with no M Ports)
	or X'01'	(Concentrator=an FDDI node with M Ports)
Mac_Ct	X'00'-X'FF'	(The number of MACs physically present in the station.)
NonMaster_Ct	X'00'-X'02'	(The number of A, B and S Ports physically present in the station.)
Master_Ct	X'00'-X'FF	(The number of M Ports physically present in the station.)

NOTE 9 – When one or more Ports or MACs physically instantiated in the MIB are not physically present, the Mac_Ct, NonMaster_Ct and Master_Ct will differ from the corresponding MIB attributes (fddiSMTMAC–Ct, fddiSMTNonMaster–Ct and fddiSMTMaster–Ct).

7.3.1.3 Station state

Parameter_Type	X'00 03'	
Parameter_Length	X'00 04'	
Parameter_Value		
Pad	X'00 00'	
Topology	X'xx'	(The following bits are set to indicate one or a combination of station topology conditions:)
	bit 0:	Peer Wrap : set when the station is in a peer wrap condition (see fddiSMTPeerWrapFlag).
	bit 1:	Unattached Concentrator: set when a concentrator has no active A, B, or S Port
	bit 2:	Twisted Ring: set when an A–A connection is detected in the station
	bit 3:	Twisted Ring: set when a B–B connection is detected in the station

	bit 4:	Rooted Station: set when the station does not have an active A, B or S Port in tree mode
	bit 5:	Status reporting: set if SRF protocol is enabled (fddiSMTStatRptPolicy)
	bit 6:	Synchronous service: set if the synchronous bandwidth allocation is greater than zero for any Path (fddiPATHSbaOverhead)
DuplAddress	X'xx'	(The following bits are set to indicate one or a combination of duplicate address conditions:)
	bit 0:	My duplicate: the value of fddiMACDA_Flag for this MAC)
	bit 1:	My UNA duplicate: the value of fddiMACUNDA_Flag for this MAC)

The topology bits of the station state SMT_Parameter in the Neighbour Information Frames provide information that allows any station on the ring to infer the global topology of the ring. If a station observes any NIF with the Peer Wrap condition bit set, it may infer that the network is wrapped on the trunk ring. If a station observes any NIF with the Unattached Concentrator bit set, it may infer that the network is comprised of a single tree with no trunk ring. Similarly, observing any NIF with a Twisted Ring bit set means that the primary and secondary rings have been crossed on the trunk ring. Observing any NIF with the Rooted Station bit set means that this ring cannot be part of a Master-Slave loop because at least one station or concentrator on the ring is either unattached or is attached to the trunk ring.

7.3.1.4 MsgTimeStamp

Parameter_Type	X'00 04'	
Parameter_Length	X'00 08'	
Parameter_Value		
MsgTimeStamp	X'xx ... xx' (8 octets)	
Reference	MsgTimeStamp:	fddiSMTTimeStamp

7.3.1.5 Station policies

Parameter_Type	X'00 05'	
Parameter_Length	X'00 04'	
Parameter_Value		
ConfigPolicy	X'xx xx'	(fddiSMTConfigPolicy)
	bit 0:	configurationhold
ConnectionPolicy	X'xx xx'	(A bit string, each appropriate bit is set)
	bit 0:	rejectA–A
	bit 1:	rejectB–B
	bit 2:	rejectS–S
	bit 3:	rejectA–M
	bit 4:	rejectB–M
	bit 5:	rejectA–S
	bit 6:	rejectB–S
	bit 7:	rejectA–B
	bit 8:	rejectB–A
	bit 9:	rejectS–A
	bit 10:	rejectS–B
	bit 11:	rejectS–M
	bit 12:	rejectM–A
	bit 13:	rejectM–B
	bit 14:	rejectM–S
	bit 15:	rejectM–M
Reference	ConfigPolicy:	fddiSMTConfigPolicy
	ConnectionPolicy:	fddiSMTConnectionPolicy

7.3.1.6 Path latency contribution/per ring OPTIONAL

Parameter_Type	X'00 06'	
Parameter_Length	X'00 08'	
Parameter_Value		
PORTOutIndex	X'xx xx'	(Ring is identified by Output Port index)
PathLatencyRing1	X'xx xx'	(In octets)
PORTOutIndex	X'xx xx'	(Dual attach)
	or X'00 00'	(Single attach)
PathLatencyRing2	X'xx xx'	(Dual attach)
	or X'00 00'	(Single attach)

NOTE 10 – This is the latency contribution of the Paths within the station. It is not a MIB attribute and is optional. The latency is given in octet units. The Port indices are those used to form the Path descriptor SMT_Parameter, X'00 08', that is described below.

7.3.1.7 MAC neighbours

Parameter_Type	X'00 07'	
Parameter_Length	X'00 10'	
Parameter_Value		
MIB_Index	X'xx xx'	(Value of fddiMACIndex)
MAC_Index	X'xx xx'	(See Path descriptor for MAC_Index definition)
UNA	X'xx ... xx' (6 octets)	(Individual address)
DNA	X'xx ... xx' (6 octets)	(Individual address)

Reference	MAC_Index:	As used in the Path Descriptor SMT_Parameter X'00 08'
	UNA:	fddiMACUpstreamNbr
	DNA:	fddiMACDownstreamNbr

7.3.1.8 Path descriptor

Parameter_Type	X'00 08'	
Parameter_Length	variable – 8 octets per Port, 8 octets per MAC	
Parameter_Value		
Port records:		
One per Port physically present in station, ordered by Port resource index.		
MIB_Index	X'xx xx'	(Value of fddiPORTIndex)
PortType	X'00'	(A)
	or X'01'	(B)
	or X'02'	(S)
	or X'03'	(M)
ConnectState	X'00'	(Disabled)
	or X'01'	(Connecting)
	or X'02'	(Standby)
	or X'03'	(Active)
RemotePORTType	X'00'	(A)
	or X'01'	(B)
	or X'02'	(S)
	or X'03'	(M)
	or X'04'	(Unknown)
RemoteMAC	X'00'	(None present)
	or X'01'	(Remote MAC indicated)
ConResourceIndex	X'00 01' – X'FF FF'	(Index (connection) of Port or MAC on Path)
MAC records:		
One per MAC physically present in station, ordered by MAC resource index.		
MAC_Addr	X'xx xx xx xx xx xx'	(MAC address associated with the resource index)
ConResourceIndex	X'00 01' – X'FF FF'	(Index (connection) of Port or MAC)

Reference	PortType:	fddiPORTMyType
	ConnectState:	fddiPORTConnectState
	RemotePortType:	fddiPORTNeighbourType
	RemoteMAC:	fddiPORTMACIndicated.r-val9

A station's topology (the internal arrangement of its MAC and Port components) is represented by a matrix as shown in table 2.

Table 2 – Station topology matrix

Path Descriptor		INPUTS						
		1:Port1	2:Port2	3:Port3	4:Port4	5:Mac1	6:Mac2	7:Mac3
O	1:Port1	0	0	0	0	0	1	0
U	2:Port2	0	0	0	0	1	0	0
T	3:Port3	0	0	0	0	0	0	1
P	4:Port4	1	0	0	0	0	0	0
U	5:Mac1	0	0	0	1	0	0	0
T	6:Mac2	0	1	0	0	0	0	0
S	7:Mac3	0	0	1	0	0	0	0

The OUTPUT of a MAC is its PH_DATA.request interface, the INPUT of a MAC is its PH_DATA.indication interface. The OUTPUT of a PHY is its PH_DATA.indication interface, the INPUT of a PHY is its PH_DATA.request interface. The matrix specifies the connections of OUTPUTS (rows) to INPUTS (columns). For example, in the matrix shown in table 2, the OUTPUT of Port2 is connected to the input of Mac1. Legal FDDI station topologies may have at most one non-zero entry per row or column.

A convention has been chosen for the SMT configuration response frame that orders the rows of the matrix by the Ports and then the MACs. A Port or MAC record, respectively, is associated with each row. Within these records, a connected resource (ConResource Index) identifies the non-zero entry for that row. This (ConResourceIndex) may be that of a MAC or of a Port.

All physically present MAC and Port entities regardless of connect state are represented in the Path descriptor. The encoding of the ConResourceIndex is as follows: for a Port or a MAC that is not part of any token Path, the ConResourceIndex is that of the entity itself, for a Port or a MAC that is part of an active token Path, the ConResourceIndex is that of the next active downstream entity in the token Path.

A sequential indexing is used to label the Ports (the range is 1 to *n* where *n* is the number of Ports physically present) and the MACs (the range for the MAC indices is *n* + 1 to *n* + *m*, where *m* is the number of MACs physically present). The indices are associated with physical components, and shall only vary when Ports or MACs are physically added or removed from the station. This index assignment is different from the indexing used to identify the Port and MAC object instances in the MIB. The Path Descriptor/SIF frame indices are consecutive with the MAC indices beginning at the last Port index plus one. The Port and MAC MIB identifiers are independent of this scheme.

A possible indexing scheme could be constructed as follows: If a station supports a dual attachment connection, the Port entity that is associated with (via PMD) the MIC that contains Primary In can be designated Port1 (index=1). The other Port of the dual attachment connection can be designated Port2 (index=2). Additional Ports in that station (concentrator) can be numbered sequentially in the order of token flow through those Ports, Port 3 (index=3), Port4 (index=4), etc. MAC entities can also be assigned identifiers sequentially, starting with the MAC(s) associated with the dual attachment connection (i.e. Mac1 (index =5), Mac2 (index=6), Mac3 (index=7), etc.).

It is an implementation issue as to how the indices are assigned within the guidelines given above. The mapping between the Path Descriptor Index and the MIB fddiMACIndex or fddiPORTIndex is supplied within the parameters contained in the SIF frames. It can be used to relate the information contained in the SIFs to the appropriate MIB object.

7.3.1.9 MAC status

One SMT_Parameter entry per MAC being reported. All MACs in a station shall be reported.

Parameter_Type		X' 00 09'	
Parameter_Length		X'00 28' octets	
Parameter_Value			
	MIB_Index	X'xx xx'	(Value of fddiMACIndex)
	MAC_Index	X'xx xx'	
	T_Req	X'xx ... xx' (4 octets)	
	T_Neg	X'xx ... xx' (4 octets)	
	T_Max	X'xx ... xx' (4 octets)	
	TVXValue	X'xx ... xx' (4 octets)	
	T_Min	X'xx ... xx' (4 octets)	
	SBA	X'xx ... xx' (4 octets)	
	Frame_Ct	X'xx ... xx' (4 octets)	
	Error_Ct	X'xx ... xx' (4 octets)	
	Lost_Ct	X'xx ... xx' (4 octets)	
Reference	MAC_Index:	As in the PathDescriptor SMT_Parameter X'00 08'	
	T_Req:	fddiMACT–Req	
	T_Neg:	fddiMACT–Neg	
	T_Max:	fddiMACT–Max	
	TVXValue:	fddiMACTVXValue	
	T_Min see MAC		
	SBA:	fddiPATHSbaPayload	
	Frame_Ct:	fddiMACFrame–Ct	
	Error_Ct:	fddiMACError–Ct	
	Lost_Ct:	fddiMACLost_Ct	

7.3.1.10 Port Link Error Rate Monitoring (LER) status

Parameter_Type		X'00 0A'	
Parameter_Length		X'00 10'	
Parameter_Value			
	MIB_Index	X'xx xx'	(Value of fddiPORTIndex)
	PORT_Index	X'00 01'-X'FF FF'	
	Pad	X'00'	
	Ler_Cutoff	X'04'-X'0F'	Default X'07'
	Ler_Alarm	X'04'-X'0F'	Default X'08'
	Ler_Estimate	X'04'-X'0F'	
	Lem_Reject_Ct	X'00 00 00 00'-X'FF FF FF FF'	
	Lem_Ct	X'00 00 00 00'-X'FF FF FF FF'	
Reference	PORT_Index:	As in the PathDescriptor SMT_Parameter X'00 08'	
	Ler_Cutoff:	fddiPORTLer–Cutoff	
	Ler_Alarm:	fddiPORTLer–Alarm	
	Ler_Estimate:	fddiPORTLer–Estimate	
	Lem_Reject_Ct:	fddiPORTLem–Reject–Ct	
	Lem_Ct:	fddiPORTLem–Ct	

7.3.1.11 MAC frame counters

Parameter_Type		X'00 0B'	
Parameter_Length		X'00 0C'	
Parameter_Value			
	MIB_Index	X'xx xx'	(Value of fddiMACIndex)
	MAC_Index	X'xx xx'	
	Copied_Ct	X'xx ... xx' (4 octets)	
	Transmit_Ct	X'xx ... xx' (4 octets)	
Reference	MAC_Index:	As in the PathDescriptor SMT_Parameter X'00 08'	
	Copied_Ct:	fddiMACCopied_Ct	
	Transmit_Ct:	fddiMACTransmit–Ct	

7.3.1.12 MAC frame not copied count OPTIONAL

Parameter_Type	X'00 0C'		
Parameter_Length	X'00 08'		
Parameter_Value			
MIB_Index	X'xx xx'	(Value of fddiMACIndex)	
MAC_Index	X'xx xx'		
NotCopied_Ct	X'xx ... xx' (4 octets)		
Reference	MAC_Index:	As in the PathDescriptor SMT_Parameter X'00 08'	
	NotCopied_Ct:	fddiMACNotCopied–Ct	

7.3.1.13 MAC priority values OPTIONAL

Parameter_Type	X'00 0D'		
Parameter_Length	X'00 20'		
Parameter_Value			
MIB_Index	X'xx xx'	(Value of fddiMACIndex)	
MAC_Index	X'xx xx'		
T_Pri0	X'xx ... xx' (4 octets)		
T_Pri1	X'xx ... xx' (4 octets)		
T_Pri2	X'xx ... xx' (4 octets)		
T_Pri3	X'xx ... xx' (4 octets)		
T_Pri4	X'xx ... xx' (4 octets)		
T_Pri5	X'xx ... xx' (4 octets)		
T_Pri6	X'xx ... xx' (4 octets)		
Reference	MAC_Index:	As in the PathDescriptor SMT_Parameter X'00 08'	
	T_Pri(n):	fddiMACT–Pri(n)	

7.3.1.14 Port EB (Elasticity Buffer) status OPTIONAL

Parameter_Type	X'00 0E'		
Parameter_Length	X'00 08'		
Parameter_Value			
MIB_Index	X'xx xx'	(Value of fddiPORTIndex)	
PORT_Index	X'00 00' – X'FF FF'		
EbErrorCt	X'00 00 00 00 – X'FF FF FF FF'		
Reference	PORT_Index:	As in the PathDescriptor SMT_Parameter X'00 08'	
	EbErrorCt:	fddiPORTEBError-Ct.	

7.3.1.15 Manufacturer field OPTIONAL

Parameter_Type	X'00 0F'		
Parameter_Length	X'00 20'		
Parameter_Value			
manuf_OUI	X'xx xx xx'	(the manufacturer's Organizationally Unique Identifier from IEEE, with I/G bit zero)	
manuf_data	X'xx ... xx' (29 octets)	(manufacturer defined data)	
Reference	manuf_OUI, manuf_data:	fddiSMTManufacturerData	

7.3.1.16 User field

Parameter_Type	X'00 10'		
Parameter_Length	X'00 20'		
Parameter_Value			
user_data	X'xx ... xx' (32 octets)		
Reference	user_data:	fddiSMTUserData	

7.3.1.17 ECHO data

Parameter_Type	X'00 11'		
Parameter_Length	X'00 00' – X'11 66'		
Parameter_Value			
echo_data	user defined		

7.3.1.18 Reason code

Parameter_Type	X'00 12'	
Parameter_Length	X'00 04'	
Parameter_Value		
reason		
	X'00 00 00 01'	(FrameClassNotSupported)
	or X'00 00 00 02'	(FrameVersionNotSupported)
	or X'00 00 00 03'	(Success)
	or X'00 00 00 04'	(BadSetCount)
	or X'00 00 00 05'	(IllegalOperation)
	or X'00 00 00 06'	(NoParameter)
	or X'00 00 00 08'	(OutOfRange)
	or X'00 00 00 09'	(NotAuthorized)
	or X'00 00 00 0A'	(LengthError)
	or X'00 00 00 0B'	(FrameTooLong)
	or X'00 00 00 0D'	(SBADenied)

7.3.1.19 Rejected frame beginning

Parameter_Type	X'00 13'	
Parameter_Length	Variable	
Parameter_Value		
Pad	X'00 00 00'	
rejected_frame_beginning		(First *n* octets of the rejected frame. This shall begin with the MAC FC field. The rejected_frame_beginning field shall be truncated, if necessary, to fit in a maximum length SMT frame or to assure that the SMT_Parameter is a multiple of four bytes in length.)

7.3.1.20 SMT supported versions

Parameter_Type	X'00 14'	
Parameter_Length	Variable (length shall be a multiple of 4)	
Parameter_Value		
pad	X'00 00'	
numVersions	X'01' – X 'FF'	(the number of versions that are supported by the station)
indexOpVersion	X'01' – X 'FF'	
Version (1)	X'00 00'–X 'FF FF'	(first of a contiguous list of supported versions..note that the list may contain only one element)
Version (n)	X'00 00'–X 'FF FF'	(last supported version)
Pad	Null or X'00 00'	(shall pad at end if there are an odd number of versions)

Reference numVersions: can be derived from the MIB by subtracting fddiSMTLoVersionId from fddiSMTHiVersionId and adding one

indexOpVersion: the index or pointer to the operational version in the list of versions that follows in the frame.

7.3.1.21 Resource type OPTIONAL

Parameter_Type	X'00 15'	
Parameter_Length	X'00 04'	
Parameter_Value		
ResourceType	X'00 00 00 01'	(Synchronous bandwidth)

7.3.1.22 SBA command OPTIONAL

Parameter_Type	X'00 16'	
Parameter_Length	X'00 04'	
Parameter_Value		

SbaCommand	X'00 00 00 01'	(Request Allocation – requests allocation of synchronous bandwidth))
	or X'00 00 00 02'	(Report Allocation – causes reporting of current synchronous bandwidth allocation)
	or X'00 00 00 03'	(Change Allocation – forces a station using synchronous bandwidth to change its current allocation)

7.3.1.23 SBA payload request OPTIONAL

Parameter_Type	X'00 17'	
Parameter_Length	X'00 04'	
Parameter_Value		
SbaRequest	X'xx ... xx' (4 octets)	(Synchronous bandwidth requested for data transmission measured in bytes per 125 µs)

7.3.1.24 SBA overhead request OPTIONAL

Parameter_Type	X'00 18'	
Parameter_Length	X'00 04'	
Parameter_Value		
SbaOverheadReq	X'xx ... xx' (4 octets)	(Synchronous bandwidth requested for overhead measured in bytes per T_Neg)

7.3.1.25 SBA allocation address OPTIONAL

Parameter_Type	X'00 19'	
Parameter_Length	X'00 08'	
Parameter_Value		
Pad	X'00 00'	
SbaAllocationAddr	X'xx ... xx' (6 octets)	(Allocation address - individual or group address responsible for the allocation)

7.3.1.26 SBA category OPTIONAL

Parameter_Type	X'00 1A'	
Parameter_Length	X'00 04'	
Parameter_Value		
Category	X'00 00 00 00'	(Category not supported or not known)
	or X'00 00 00 01'	(Static allocation)
	or X'xx ... xx' (4 octets)	(Allocator defined classification for the bandwidth)

7.3.1.27 Maximum T_Neg OPTIONAL

Parameter_Type	X'00 1B'	
Parameter_Length	X'00 04'	
Parameter_Value		
MaxTneg	X'xx ... xx' (4 octets)	(The unsigned twos complement representation for the longest T_Neg acceptable for the requested synchronous service)

7.3.1.28 Minimum SBA segment size OPTIONAL

Parameter_Type	X'00 1C'	
Parameter_Length	X'00 04'	
Parameter_Value		
MinSegment	X'xx ... xx' (4 octets)	(The smallest number of bytes per frame into which the synchronous payload will be segmented, default = 1)

7.3.1.29 SBA allocatable OPTIONAL

Parameter_Type	X'00 1D'	
Parameter_Length	X'00 04'	
Parameter_Value		
SbaAllocatable	X'xx ... xx' (4 octets)	(Synchronous bandwidth available for allocation as reported by the management process. This value is measured in bytes per maximum T_Neg (see 7.3.1.27).)

7.3.1.30 Authorization OPTIONAL

Parameter_Type	X'00 21'	
Parameter_Length	X'xx xx'	(User defined length. This shall be a multiple of 4 octets to preserve 32-bit alignment)
Parameter_Value		
Authorization	X'xx … xx'	(User defined authorization field.)

7.3.1.31 ESF ID

Parameter_Type	X'FF FF'	
Parameter_Length	X'00 08'	
Parameter_Value		
Pad	X'00 00'	
ESF_ID	X'xx ... xx' (6 octets)	(ESF identifier. An IEEE 48 bit assigned ID. The ESF ID shall come from the universally administered part of the IEEE address space. The I/G bit and the U/L bit of the IEEE address shall be zero. The representation shall be the MSB (most significant bit) FDDI–MAC order.)

8 Frame-based management protocols

SMT provides a number of Frame-Based services and functions that may be used by higher-level management functions to gather information about and exercise control over the attached FDDI network.

The frame-based protocols defined in SMT provide the capabilities for the gathering of network statistics; the detection, isolation, and resolution of network faults; and the tuning of FDDI configuration and operational parameters to meet application performance and connectivity requirements.

8.1 Frame processing

Requirements that are common across SMT frame protocols are described in this clause.

Received SMT frames that do not satisfy the validity criteria specified in 7.1.7 or the length criteria specified in 7.1.2.5 shall not be processed by the SMT frame protocols.

8.1.1 Request–response protocols

A number of the SMT frame protocols are request-response frame protocols. These protocols are carried out between a single requester and one or more responders, depending on the addressing mode of the request frame.

With the exception of the NIF request generated periodically by the Neighbour Notification protocol (see 8.2) and the RAF request generated for allocation of resourses, request frames are generated only at the request of a management agent process. These request frames may be generated at any time, and to any destination or destinations. The ability of a station to generate such request frames is not required.

Unless otherwise specified in the individual protocol descriptions, the following rules apply to all SMT request-response associations:

a) The Destination Address of the response shall be the Source Address of the request.

b) The Source Address of the response shall be the fddiMACSMTAddress of the MAC that received the request.

c) The response is transmitted on the Path on which the request was received.

d) The request is interpreted according to the rules of the Version ID indicated by the request. If the indicated version is supported, the Version ID of the response shall be the Version ID of the request.

e) The Transaction ID of the response shall be the Transaction ID of the request.

f) Stations are required to respond to requests. Under 'zero-load' conditions, stations shall respond within 30 s. 'Zero-load' conditions exist when a station has empty transmit and receive queues, and when there is no network load except for the request to which the station is responding. Under other load conditions a response remains required, although no upper bound for the response time is specified.

For the purposes of the protocol descriptions, a frame that is queued for transmission is considered transmitted by the sender of the frame. The frame may be subsequently discarded before actual transmission occurs, due to errors, exceptions, or fluctuations in the operation of the FDDI network.

Certain responses, specifically request denied frames and parameter management response frames, include a reason code (see 7.3.1.18) that indicates the success or failure of the request. The definitions of specific reason codes, and the circumstances under which they are transmitted, are specified as part of the SMT frame protocols that use them.

When a response is received by a requesting station, unless otherwise specified, the transaction ID of the response will typically be used to identify and notify the management agent process that originated the request. The means by which this function is carried out are not specified.

8.1.2 Announcement protocols

Other SMT frame protocols, such as the status reporting protocol, are unacknowledged announcement protocols.

When an announcement is received by a station, typically a management agent process will be notified. The means by which this function is carried out are not specified.

8.1.3 SMT header processing

SMT frames that meet the validity and length criteria shall be processed according to the SMT frame protocols described in this and subsequent subclauses. Initially, the SMT Header is examined to identify the SMT frame protocol and version being exercised. The SMT Header and the fields that comprise it are described in 7.1.2.

The following subclauses identify exception conditions in SMT header processing. If any of these conditions are satisfied, the action specified is performed, and processing of the frame is terminated. If more than one of the exception conditions are simultaneously satisfied, then the implementer may choose which of the actions is performed. If none of the exceptions occur, frame processing proceeds according to the protocol indicated by the received frame class.

8.1.3.1 Unsupported frame class

If the received frame class is not supported, and the received frame type is 'Request,' then a request denied frame with reason code of 'FrameClassNotSupported' shall be sent in response. If the received frame class is not supported, and the received frame type is not 'Request,' then the frame shall be discarded.

8.1.3.2 Unsupported frame type

If the received frame type is undefined or not defined within the scope of the received frame class, then the frame shall be discarded.

8.1.3.3 Unsupported version ID

If the received Version ID is not supported, and the received frame type is 'Request,' then a request denied frame with reason code of 'FrameVersionNotSupported' shall be sent in response. The version ID of the request denied frame shall be the supported version ID that is numerically nearest to the version ID of the request. If the received Version ID is not supported, and the received frame type is not 'Request,' then the frame shall be discarded.

8.1.3.4 Invalid InfoField length

If the received InfoField length is longer than the remaining frame INFO field, and the received frame type is 'Request,' then a request denied frame with reason code of 'LengthError' may optionally be sent in response. If the received InfoField length is longer than the remaining frame INFO field, and the received frame type is not 'Request,' then the frame shall be discarded.

If the received InfoField length is shorter than the remaining frame INFO field, then, at the implementer's option, the frame may be further processed, up to the point indicated by the received InfoField length. Alternatively, if the received frame type is 'Request,' a request denied frame with reason code of 'LengthError' may optionally be sent in response; otherwise, the frame shall be discarded.

8.2 Neighbour Notification

This subclause describes the Neighbour Notification (NN) process to be employed on an operational FDDI logical ring. This protocol complements the duplicate address detection function of Ring Management (RMT), which operates when the logical ring is not operational.

The Neighbour Notification protocol performs the following functions:

a) It determines a MAC's logical upstream neighbour address (UNA) and its logical downstream neighbour address (DNA);

b) It detects a duplicated fddiMACSMTAddress on an operational FDDI logical ring, supplementing the duplicate address detection function provided by RMT when the ring is not operational;

c) It generates a periodic frame handshake that verifies the operation of the local MAC receive and transmit Paths, in the absence of any other traffic.

The protocol performs these functions by periodically initiating a request-response frame exchange between a MAC and its nearest downstream neighbour. The frames used are the SMT Neighbour Information Frames (see 7.2.1).

The Neighbour Notification process is performed independently for each MAC in a station.

The Neighbour Notification process is required and supports higher-level network management functions. It facilitates the detection of, isolation of, and recovery from network errors by supplying each individual MAC with logical fault domain information. The neighbour information at each MAC may also be collected by a higher-level management function for the purpose of constructing and verifying a logical ring map.

8.2.1 Neighbour information polling

In addition to their application in the Neighbour Notification protocol, Neighbour Information Frames (NIFs) may be used to retrieve neighbour information from any other station on an FDDI network. The neighbour information carried in the NIFs collected from all stations on an FDDI ring is sufficient for the construction of a logical ring map.

Excluding the operation of the Neighbour Notification process, a NIF request or NIF announcement is sent only at the request of a higher-level management entity. A station may issue NIF requests or NIF announcements at any time and to any destination.

Some of the parameters that are included in the NIFs are derived from the state of the MAC that transmits the NIF. The fddiMACSMTAddress of this MAC appears as the Source Address of the NIF.

8.2.2 Facilities

The Neighbour Notification protocol is performed by two cooperating processes. The Neighbour Notification transmitter is responsible for periodically issuing NIF request frames to the downstream neighbour, and processing NIF response frames from the downstream neighbour. The NN receiver is responsible for responding to NIF request frames from the upstream neighbour. There is one instance of the NN transmitter and one instance of the NN receiver for each MAC in a station.

The Neighbour Notification protocol defines and makes use of the following variables, timers, and signals. These facilities are used for maintaining state information, effecting state transitions, and communicating results to other management entities.

8.2.2.1 Variables

A variable may take on one value from a limited set of allowable values. A flag, or Boolean variable, may take on only two values, namely SET and CLEAR. Variables may be modified by state or transition actions in the state machines. Variables may be tested by state actions or by transition conditions. Variable values may be exported to other entities within or external to SMT.

UNA Upstream Neighbour Address. The UNA identifies the most-recently-known upstream neighbour address. This variable is exported to external management entities. If unknown, the UNA is reported as SMT_Unknown_Address.

Old_UNA Old Upstream Neighbour Address. The UNA identifies the previously-known upstream neighbour. This variable is exported to external management entities. If unknown, the Old_UNA is reported as SMT_Unknown_Address.

DNA Downstream Neighbour Address. The DNA identifies the most-recently-known downstream neighbour. This variable is exported to external management entities. If unknown, the DNA is reported as SMT_Unknown_Address.

Old_DNA Old Downstream Neighbour Address. The DNA identifies the previously-known downstream neighbour. This variable is exported to external management entities. If unknown, the Old_DNA is reported as SMT_Unknown_Address.

Dup_Addr_Test A variable that indicates the current status of the duplicate address detection function. It takes on the values:

None The duplicate address test has not been completed.

Pass The duplicate address test has completed and HAS NOT detected a duplicate address.

Fail The duplicate address test has completed and HAS detected a duplicate address.

The current value of Dup_Addr_Test is exported to other management entities, including RMT and external management authorities.

OPTION:
The value of Dup_Addr_Test may be set to Pass upon successful transmission and reception of a self-addressed (DA = SA = fddiMACSMTAddress) frame by a full-duplex MAC implementation. If so , the frame shall be an SMT frame (FC=X'41' .. X'4F' (SMT)) and shall include the SMT header. The frame type and contents are the implementer's choice.

This self-addressed frame may be sent in addition to the NIF request frame specified in Send_Actions below. Successful reception is defined as receipt of a self-addressed frame that is identical to the transmitted frame, without receive errors (reception_status = (FR_GOOD, E_r = R)).

UNDA_Flag Upstream neighbour duplicate address flag. This flag indicates that the current upstream neighbour has detected itself to be a duplicate. The default and initial value is Clear.

NN_Transaction_ID The Transaction_ID currently in use by the Neighbour Notification protocol. This value is the Transaction_ID for NIF request frames transmitted by, and is the expected Transaction_ID for NIF response frames received by, the NN transmitter process.

Successive new NN_Transaction_IDs are generated at a defined time in the Neighbour Notification protocol. A specific NN_Transaction_ID will be re-used as the Transaction_ID in successive NIF request frames until a NIF response frame with Transaction_ID matching NN_Transaction_ID is received.

Successive values of NN_Transaction_ID shall be distinct (not required across self-test or station initialization cycles) from each other and from Transaction_IDs outstanding for NIF requests sent by SMT for other purposes, but are otherwise the implementer's choice.

8.2.2.2 Timers

TNN Timer, Neighbour Notification. TNN is initialized and enabled at station initialization. It runs continuously and repeatedly expires with a period of T_Notify. Each expiration causes the signal TNN_expires to be asserted. A single TNN timer, and therefore a single TNN_expires signal, may be shared by all NN transmitter processes active in a station.

TVU Timer, Valid Upstream Neighbour. This timer is reset on each reception of a NIF request frame or NIF announcement frame from the upstream neighbour. If it reaches the timeout value, T_NN_Out, the current UNA is set to unknown.

TVD Timer, Valid Downstream Neighbour. This timer is reset on each reception of a NIF response frame from the downstream neighbour. If it reaches the timeout value, T_NN_Out, the current DNA is set to unknown.

8.2.2.3 Timer expiration values

T_Notify The interval between successive iterations of the Neighbour Notification protocol. The nominal value of T_Notify may vary over the range of 2 s to 30 s, inclusive. It is recommended that the actual interval between successive iterations be uniformly distributed over a range of ±1 s about the nominal value, to minimize the effects of the broadcast nature of the NIF request

T_Notify_Min=2 s

T_Notify_Max=30 s

T_Notify=30 s (default)

T_NN_Out The maximum interval between successive NIF receptions from the upstream and downstream neighbours to maintain neighbour information validity.

$T\_NN\_Out > M\_Max \times T\_Max + 2 \times T\_Notify\_Max = 228$ s

8.2.2.4 Signals

A signal is an indication that may cause a state change to occur.

TNN_expires This signal is generated on each expiration of the free-running TNN timer. This signal is consumed by the NN transmitter process and triggers the transmission of the NIF request frame to the downstream neighbour. A single TNN_expires signal may be shared by all active NN transmitters in a station.

UNA_change This signal is generated when a MAC UNA changes. The new address is stored in UNA, and the previous address is stored in Old_UNA.

DNA_change This signal is generated when a MAC DNA changes. The new address is stored in DNA, and the previous address is stored in Old_DNA.

8.2.3 Neighbour Notification transmitter operation

The Neighbour Notification transmitter state diagram is depicted in figure 7 and associated footnotes. The notation $Field_r$ used in the footnotes designates a frame field received by this MAC, and $Field_x$ designates a frame field to be transmitted by this MAC.

8.2.3.1 State NT0:WAIT

The NN transmitter will be in the Wait state during the intervals between successive completed Neighbour Notification protocol frame exchanges. The NN transmitter also enters this state anytime the fddiMACSMTAddress changes.

NT(n0): Address Change - A transition to the Wait state, from any NN transmitter state, occurs when the fddiMACSMTAddress changes. Dup_Addr_Test is reset to its initial value.

NT(01): TNN_expires AND SM_MA_Avail: If the TNN_expires signal is received and the ring to which the MAC is attached is operational, the Send state is entered, and a NIF request frame will be transmitted.

NT(00b): DupResponse: If a NIF response is received in the Wait state with reception_status =(FR_GOOD, Er = R, Ar = S, Cr = R | S), this fddiMACSMTAddress is duplicated on the logical ring. Dup_Addr_Test is set to Fail, the station state SMT_Parameter associated with this MAC is updated.

8.2.3.2 State NT1:SEND

The NN transmitter enters the Send state when the conditions to transmit a NIF request frame are met. Upon entry into the Send state a NIF request frame is generated and queued for transmission. While in the Send state, the NN transmitter is awaiting a response to the transmitted NIF request.

NT(10a): NormResponse - If a NIF response is received in the Send state with reception_status =(FR_GOOD, E_r = R, A_r = R, C_r = R | S) and Transaction_ID_r = NN_Transaction_ID, this fddiMACSMTAddress is not duplicated on the logical ring. Dup_Addr_Test is set to Pass, the station state SMT_Parameter associated with this MAC is updated, a new NN_Transaction_ID is generated, and optionally, the SA_r of the NIF response frame is recorded as this MAC's current DNA.

NT(10b): DupResponse - If a NIF response is received in the Send state with reception_status =(FR_GOOD, Er = R, Ar = S, C_r = R | S), this fddiMACSMTAddress is duplicated on the logical ring. Dup_Addr_Test is set to Fail, the station state SMT_Parameter associated with this MAC is updated, and optionally, the stored DNA information for this MAC is invalidated.

NT(11): TNN_expires AND SM_MA_Avail - If the TNN_expires signal is received and the ring to which the MAC is attached is operational, the Send state is re-entered, and the NIF request frame will be retransmitted.

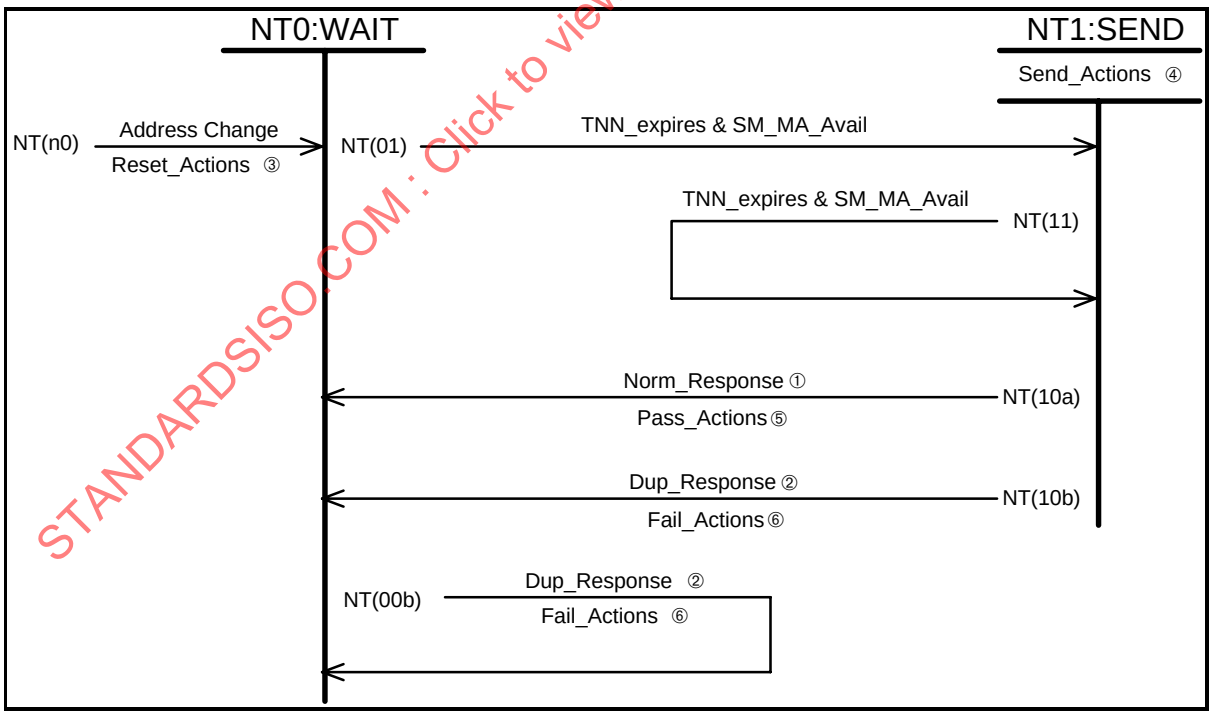


Figure 7 – Neighbour Notification transmitter state diagram (Part 1 of 3)

1. Norm_Response:
   This condition is satisfied by receipt of a NIF Response frame (see 7.2.1.3) (via SM_MA_UNITDATA.indication) with reception_status of (FR_GOOD, Er = R, Ar = R, Cr = R | S), and frame contents:

   FC_r=X'41' (SMT Info)

   DA_r = This fddiMACSMTAddress

   SMT HEADER:

   $FrameClass_r$= X'01' (NIF)

   $FrameType_r$ = X'03' (Response)

   $VersionID_r$ = X'00 01' (Constant)

   $Transaction\_ID_r$ = NN_Transaction_ID

   Receipt of this frame in the Send State with the specified reception_status and Transaction_ID = NN_Transaction_ID indicates that the fddiMACSMTAddress is not duplicated.

2. Dup_Response:
   This condition is satisfied by receipt of a NIF Response frame (see 7.2.1.3) (via SM_MA_UNITDATA.indication) with reception_status of (FR_GOOD, Er = R, Ar = S, Cr = R | S), and frame contents:

   FC_r=X'41' (SMT Info)

   DAr = This fddiMACSMTAddress

   SMT HEADER:

   $FrameClass_r$= X'01' (NIF)

   $FrameType_r$ = X'03' (Response)

   $VersionID_r$ = X'00 01' (Constant)

   Receipt of this frame in either the Send State or the Wait State with the specified reception_status indicates that the fddiMACSMTAddress is duplicated.

3. Reset_Actions:
   Dup_Addr_Test := None;
   NN_Transaction_ID := New Transaction_ID;

4. Send_Actions:
   Queues a NIF Request (see 7.2.1.2) frame for transmission by this MAC with asynchronous service (via SM_MA_UNITDATA.request).

   The frame includes the following information:

   FC_X=X'4F' (SMT NSA)

   DAx = X'FF FF FF FF FF FF' (Broadcast)

   SAx = This fddiMACSMTAddress

   SMT HEADER:

   FrameClassx = X'01' (NIF)

   $FrameType_X$ = X'02' (Request)

   $VersionID_X$ = X'00 01' (Constant)

   $Transaction\_ID_X$ = NN_Transaction_ID

   $StationID_X$ = This SMTStationID

5.Pass_Actions:
   Dup_Addr_Test := Pass;
   NN_Transaction_ID := New Transaction_ID;
   reset(TVD);
   IF $SA_r \neq DNA$
   THEN
   BEGIN
   IF DNA _ SMT_Unknown_Address
   THEN Old_DNA := DNA;
   DNA := SA_r;
   SIGNAL DNA_Change
   END

6. Fail_Actions:
   Dup_Addr_Test := Fail;
   IF ($SA_r \neq DNA$) AND ($Transaction\ ID_r$ = NN_Transaction_ID)
   THEN
   BEGIN
   IF DNA _ SMT_Unknown_Address
   THEN Old_DNA := DNA;
   DNA := SA_r;
   SIGNAL DNA_Change
   END

Figure 7 – Neighbour Notification transmitter state diagram (Part 2 of 2)

8.2.4 Neighbour Notification receiver operation

The Neighbour Notification receiver is described in pseudo-code as follows:

```
WHILE TRUE DO
      BEGIN
            IF NIF_Received [footnote] 1
                  THEN Respond_Actions [footnote] 2;
            IF (TVU > T_NN_Out) AND (UNA _ SMT_Unknown_Address)
                  THEN
                        BEGIN
                              Old_UNA := UNA;
                              UNA := SMT_Unknown_Address;
                              SIGNAL UNA_Change;
                              CLEAR UNDA_Flag
                        END;
            IF (TVD > T_NN_Out) AND (DNA _ SMT_Unknown_Address)
                  THEN
                        BEGIN
                              Old_DNA := DNA;
                              DNA := SMT_Unknown_Address;
                              SIGNAL DNA_Change
                        END
      END
```

The following footnotes are associated with Neighbour Notification receiver operation.

1. NIF_Received:
   This condition is satisfied by receipt of a NIF Announcement (see 7.2.1.1) or a NIF Request (see 7.2.1.2) frame (via SM_MA_UNITDATA.Indication) with reception_status of (FR_GOOD, E_r=R, A_r=R, C_r=R), and frame contents:
   FC_r=X'4F' {SMT NSA}
   DA_r=X'FF FF FF FF FF FF' {Broadcast}
   SMT HEADER:
   $FrameClass_r$ = X'01' {NIF}
   $FrameType_r$ = X'01' (Announcement) | X'02' {Request}
   $VersionID_r$ = X'00 01' {constant}
   This reception_status and frame contents indicate that the sender of the frame is this MAC's upstream neighbour.
2. Respond_Actions:

```
RESET TVU;
IF SAr ≠ UNA
      THEN
            BEGIN
                  IF UNA ≠ SMT_Unknown_Address
                        THEN Old_UNA := UNA;
                  UNA := SAr;
                  SIGNAL UNA_Change
            END;
IF StationStater.DuplAddress.MyDuplicate = SET
      THEN
            SET UNDA_Flag
      ELSE
            CLEAR UNDA_Flag;
IF FrameTyper = 2 { Request }
      THEN
            BEGIN
                  Construct NIF_Response; {see 7.2.1.3}
                  QUEUE(NIF_Response) { via this
                        MAC:SM_MA_UNITDATA.request(Asynchronous) }
            END
```

8.3 Status Report protocol

8.3.1 Overview

A station performs the Status Report (SR) protocol to periodically announce Station Status that is useful in managing an FDDI ring. This status information is carried in Status Report frames (see 7.2.7)

In reporting status, the protocol considers two different types of status: conditions and events.

8.3.1.1 Conditions

Conditions are station states that are reported to the SRF multicast address (SMT-SRF-DA) (see 7.1.6) as long as the condition remains asserted. new condition assertions and new condition deassertions are reported within a rate-limiting interval (T_Limit). While conditions remain asserted, in the absence of new condition transitions or event occurrences, they are continuously reported with a geometrically increasing report interval (SRThreshold). Each condition report contains status information collected at the time that the Status Report frame is generated.

Some conditions are asserted when a particular counter ratio is exceeded. These ratios are calculated at a sample interval that is implementer defined. The interval may be either fixed or variable. At each interval, the necessary ratios are calculated and the corresponding conditions are asserted if the ratio exceeds the threshold. A threshold of zero causes any increment in the error counters to generate an event. Defined conditions include:

Frame error condition	Active when fddiMACFrameErrorRatio over an implementation-specific sample interval is greater than or equal to fddiMACFrameErrorThreshold.
LER condition	Active when a Port LER_Estimate is greater than or equal to LER_Alarm (see 9.4.7).
Duplicate Address condition	Active when a MAC detects itself or its UNA to be a duplicate address (see 8.2).
Peer Wrap condition	Active when a dual-attachment node is wrapped with a Peer mode connection (PeerWrapFlag, see 9.7.2.4.4).
Hold condition	Active when a station enters a 'holding-prm' or 'holding-sec' state (see 9.5).
NotCopied condition	OPTIONAL Active when the fddiMACFrameNotCopiedRatio over an implementation-specific sample interval is greater than or equal to fddiMACFrameNotCopiedThreshold.
EB error condition	OPTIONAL Active when any fddiPORTEBError-Ct has incremented over an implementation-specific sample interval.
Vendor-Specific conditions	OPTIONAL The semantics of a vendor-specific condition is defined by its owner, as identified by a Organizationally-Unique-Identifier (OUI) that is part of the condition encoding. Any number of OUI-specific conditions may be supported by an SMT implementation. In addition, multiple vendor-specific conditions may be included within the scope of a single OUI, and are distinguished in a vendor-specific fashion.

8.3.1.2 Events

Events are occurrences which are reported to the SRF multicast address (SMT-SRF-DA) (see 7.1.6). Event occurrences are reported in Status Report frames within a rate-limiting interval (T_Limit). Each event report contains status information collected at the time that the Status Report frame is generated. Defined events include:

MAC Path change	Generated when the value of current Path changes from or to either Primary Path or Secondary Path for any MAC in a station (see 9.7.2.4.4).
Port Path change	Generated when the value of current Path changes for any Port in a station (see 9.7.2.4.4).
MAC neighbour change	Generated when the Neighbour Notification protocol detects a UNA or DNA change for any MAC in a station (see 8.2).
Undesirable connection	Generated when an undesirable connection (see 5.2.4) is detected for any Port in a station.

Vendor-Specific events OPTIONAL The semantics of a vendor-specific event is defined by its owner, as identified by a Organizationally-Unique-Identifier (OUI) that is part of the event encoding. Any number of OUI-specific events may be supported by an SMT implementation. In addition, multiple vendor-specific events may be included within the scope of a single OUI, and are distinguished in a vendor-specific fashion.

8.3.1.3 Operation

The Status Report protocol is performed by the Status Report transmitter. This process is responsible for accumulating event and condition information, constructing the Status Report frame, and transmitting the Status Report frame. There is one instance of the Status Report transmitter per station.

When conditions are active or when events occur, the Status Report transmitter registers that fact by setting the Report Required flag (ReportRequired) corresponding to the source and type of the condition or event. Subsequently, the Status Report transmitter constructs a Status Report frame with the frame parameters corresponding to all conditions and events for which ReportRequired is SET. The state of SM_MA_Avail from RMT (see 10.2.1.1) is tested at each MAC and the frame is queued to those MACs for which SM_MA_Avail is SET. If the frame is queued to one or more MACs, it is considered to be successfully transmitted.

8.3.2 Facilities

The Status Report protocol defines and makes use of the following variables, timers, and signals. These facilities are used for maintaining state information and effecting state transitions.

8.3.2.1 Variables

A variable may take on one value from a limited set of allowable values. A flag, or Boolean variable, may take on two values, SET and CLEAR. Variables may be modified by state or transition actions in the state machines. Variables may be tested by state actions or by transition conditions. Variable values may be exported to other entities within or external to SMT.

SRThreshold The interval between successive Status Report transmissions in the absence of new events or condition transitions. The value of SRThreshold is modified by Status Report transmitter actions. The minimum and initial value of SRThreshold is 2 s. The maximum value of SRThreshold is 32 s.

RT_Flag Report Transmitted flag. This flag is set if a given Status Report frame is successfully queued for transmission by at least one MAC in the station. One instance of this flag exists within the Status Report transmitter.

ReportRequired Report Required flag. This flag is set if a given event, condition assertion, or condition deassertion, for any source (i.e. any instance of any managed object in the station), is to be reported in the next Status Report frame. One instance of this flag exists for each event and condition type for each instance of each managed object.

MultipleOccurrence Multiple Occurrence flag. This flag is set if a given event for a given source occurs more than once in the interval between successive Status Report frame transmissions. One instance of this flag exists for each event type for each instance of each managed object supported by the station.

ConditionState ConditionState flag. This flag is set if a given condition for a given source is currently asserted. One instance of this flag exists for each condition type for each instance of each managed object supported by the station.

SR_Enable StatusReportEnable flag (fddiSMTStatRptPolicy). This flag indicates that the Status Report protocol is enabled for this station. The default and initial value is SET.

TimeStamp TimeStamp is defined to allow rate calculations from counter values. The timestamp is 64 bits to provide a large enough range to avoid "wrapping". The resolution of the timestamp is 80 ns (in the least significant bit). Stations with coarser timers may increment their timestamps by *N* every *N**80 ns where *N* is at most 100 000. The station's timestamp (fddiSMTTimeStamp) is an example of an attribute that may be reset on power change, station self-test, or Path-test.

TransitionTimeStamp This variable reflects the time of the most recent event occurrence, condition assertion, or condition deassertion in the station. TransitionTimeStamp is updated by storing the value of TimeStamp.

8.3.2.2 Timers

TSR Timer, Status Report. This timer paces Status Report frame generation and transmission.

8.3.2.3 Timer expiration values

T_Limit The rate-limiting interval for the Status Report protocol. Status Report frames will not be generated more often than once per T_Limit.

T_Limit = 2 s

8.3.2.4 Signals

A signal is an indication that may cause a state change to occur. The three signals defined below all represent occurrences that are reported by the Status Report protocol. Each signal is comprised of several factors, each of which represents one of the defined conditions or events. Each signal carries with it the necessary information to identify type and source of condition or event; and its source, including its type and index.

Condition_Asserted This signal is generated whenever any of the following occur:

Frame error condition becomes true. This also results in fddiMACFrameErrorFlag being SET. The associated Status Report frame SMT_Parameter is defined by fddiMACFrameErrorCondition (see 6.4.5.6.2).

OR

Duplicate Address condition becomes true. The associated Status Report frame SMT_Parameter is defined by fddiMACDuplicateAddressCondition (see 6.4.5.6.2).

OR

LER condition becomes true. This also results in fddiPORTLerFlag being SET. The associated Status Report frame SMT_Parameter is defined by fddiPORTLerCondition (see 6.4.5.6.4).

OR

Hold condition becomes true. The associated Status Report frame SMT_Parameter is defined by fddiSMTHoldCondition (see 6.4.5.6.1).

OR

Peer Wrap condition becomes true.The associated Status Report frame SMT_Parameter is defined by fddiSMTPeerWrapCondition (see 6.4.5.6.1).

OR (OPTIONAL)

NotCopied condition becomes true. This also results in fddiMACNotCopiedFlag being SET.The associated Status Report frame SMT_Parameter is defined by fddiMACNotCopiedCondition (see 6.4.5.6.2).

OR (OPTIONAL)

EB error condition becomes true. The associated Status Report frame SMT_Parameter is defined by fddiPORTEBErrorCondition (see 6.4.5.6.4).

OR (OPTIONAL)

Any Vendor-Specific condition becomes true. The associated Status Report frame SMT_Parameters are defined by fddiSMTVendorNotification (see 6.4.5.6.1), fddiMACVendorNotification (see 6.4.5.6.2), fddiPATHVendorNotification (see 6.4.5.6.3), and fddiPORTVendorNotification (see 6.4.5.6.4).

Condition_Deasserted This signal is generated whenever any of the following occur for a managed object for which the corresponding condition has previously been asserted via the Condition_Asserted signal:

Frame error condition becomes false for any MAC in the station. This also results in FrameErrorCondition (fddiMACFrameErrorFlag) being CLEAR for the affected MAC.

OR

Both Duplicate Address conditions become false for any MAC in the station.

OR

LER condition becomes false for any Port in the station. This also results in LerConditionState (fddiPORTLerFlag) being CLEAR for the affected Port.

OR

Hold condition becomes false.

OR

The Peer Wrap condition becomes false.

OR (OPTIONAL)

NotCopied condition becomes false for any MAC in the station. This also results in NotCopiedCondition (fddiMACNotCopiedFlag) being CLEAR for the affected MAC.

OR (OPTIONAL)

EB error condition becomes false for any Port in the station.

OR (OPTIONAL)

Any Vendor-Specific condition becomes false.

Event_Occurred This signal is generated whenever any of the following occur:

MAC Path change occurs for any MAC in the station. The associated Status Report frame SMT_Parameter is defined by fddiMACPathChangeEvent (see 6.4.5.6.2).

OR

Port Path change occurs for any Port in the station. The associated Status Report frame SMT_Parameter is defined by fddiPORTPathChangeEvent (see 6.4.5.6.4).

OR

MAC Neighbour change occurs for any MAC in the station. The associated Status Report frame SMT_Parameter is defined by fddiMACNeighbourChangeEvent (see 6.4.5.6.2).

OR

Undesirable connection occurs for any Port in the station. Undesirable connections include A-A, B-B, A-S, B-S, S-A, S-B, and M-M (MyType-NeighbourType). The associated Status Report frame SMT_Parameter is defined by fddiMACUndesiredConnectionAttemptEvent (see 6.4.5.6.4).

OR (OPTIONAL)

Any Vendor-Specific event occurs. The associated Status Report frame SMT_Parameters are defined by fddiSMTVendorNotification (see 6.4.5.6.1), fddiMACVendorNotification (see 6.4.5.6.2), fddiPATHVendorNotification (see 6.4.5.6.3), and fddiPORTVendorNotification (see 6.4.5.6.4).

8.3.3 Status Report transmitter operation

The Status Report transmitter state diagram is depicted in figure 8 and associated footnotes.

8.3.3.1 State SR0:WAIT

The Status Report transmitter will be in the Wait state during the intervals between Event and condition assertions and successive Status Report frame transmissions. Upon entry to this state, TSR is RESET.

SR(n0a): Station Reset - A transition to the Wait state, from any Status Report transmitter state, occurs on station reset. SRThreshold is reset to its initial value and all ReportRequireds are cleared.

SR(00b): Condition_Asserted AND TSR ≥ T_Limit: If a condition becomes asserted more than T_Limit after the last Status Report frame was transmitted, the new condition will be included in the next report, and the frame will be constructed and sent. In this case, SRThreshold is reset to its initial value.

SR(00c): Condition_Deasserted AND TSR ≥ T_Limit: If a condition becomes deasserted more than T_Limit after the last Status Report frame was transmitted, the conditionState corresponding to that condition will be updated to reflect the deassertion, and the frame will be constructed and sent.

SR(00d): Event_Occurred AND TSR ≥ T_Limit: If an event occurs more than T_Limit after the last Status Report frame was transmitted, the new Event will be included in the next report, and the frame will be constructed and sent.

SR(00e): Any ReportRequired AND TSR ≥ SRThreshold: If there is status information to report and the next reporting interval has elapsed, the new reporting interval is calculated and a Status Report frame is constructed and sent.

SR(01a): Condition_Asserted AND TSR < T_Limit: If a condition becomes asserted less than T_Limit after the last Status Report frame was transmitted, the new condition is included in the next report, and a transition to SR1:HOLDOFF will occur, deferring transmission of the frame until T_Limit has expired. In this case, SRThreshold is reset to its initial value.

SR(01b): Condition_Deasserted AND TSR < T_Limit: If a condition becomes deasserted less than T_Limit after the last Status Report frame was transmitted, the conditionState corresponding to that condition is updated to reflect the deassertion, and a transition to SR1:HOLDOFF will occur, deferring transmission of the frame until T_Limit has expired.

SR(01c): Event_Occurred AND TSR < T_Limit: If an event occurs less than T_Limit after the last Status Report frame was transmitted, the new event is included in the next report, and a transition to SR1:HOLDOFF will occur, deferring transmission of the frame until T_Limit has expired.

SR(02): NOT SR_Enable: The status transmitter enters the Disabled state when status reporting has been disabled by clearing the SR_Enable flag.

8.3.3.2 State SR1:HOLDOFF

The status transmitter is in the Holdoff state when transmission of the next Status Report frame is deferred due to T_Limit.

SR(10b): TSR ≥ T_Limit: T_Limit has elapsed since the last Status Report frame transmission, and additional events and/or conditions have occurred in the interim. The Status Report frame is constructed and sent.

SR(11a): Condition_Asserted: A condition has become asserted while waiting to send the next Status Report frame. The new condition is included in the next report.

SR(11b): Condition_Deasserted: A condition has become deasserted while waiting to send the Status Report frame. The corresponding conditionState SMT_Parameter will be updated in the next report.

SR(11c): Event_Occurred: An event has occurred while waiting to send the Status Report frame. The event SMT_Parameter is included in the next report.

SR(12): NOT SR_Enable: The status transmitter enters the disabled state when status reporting has been disabled by clearing the SR_Enable flag.

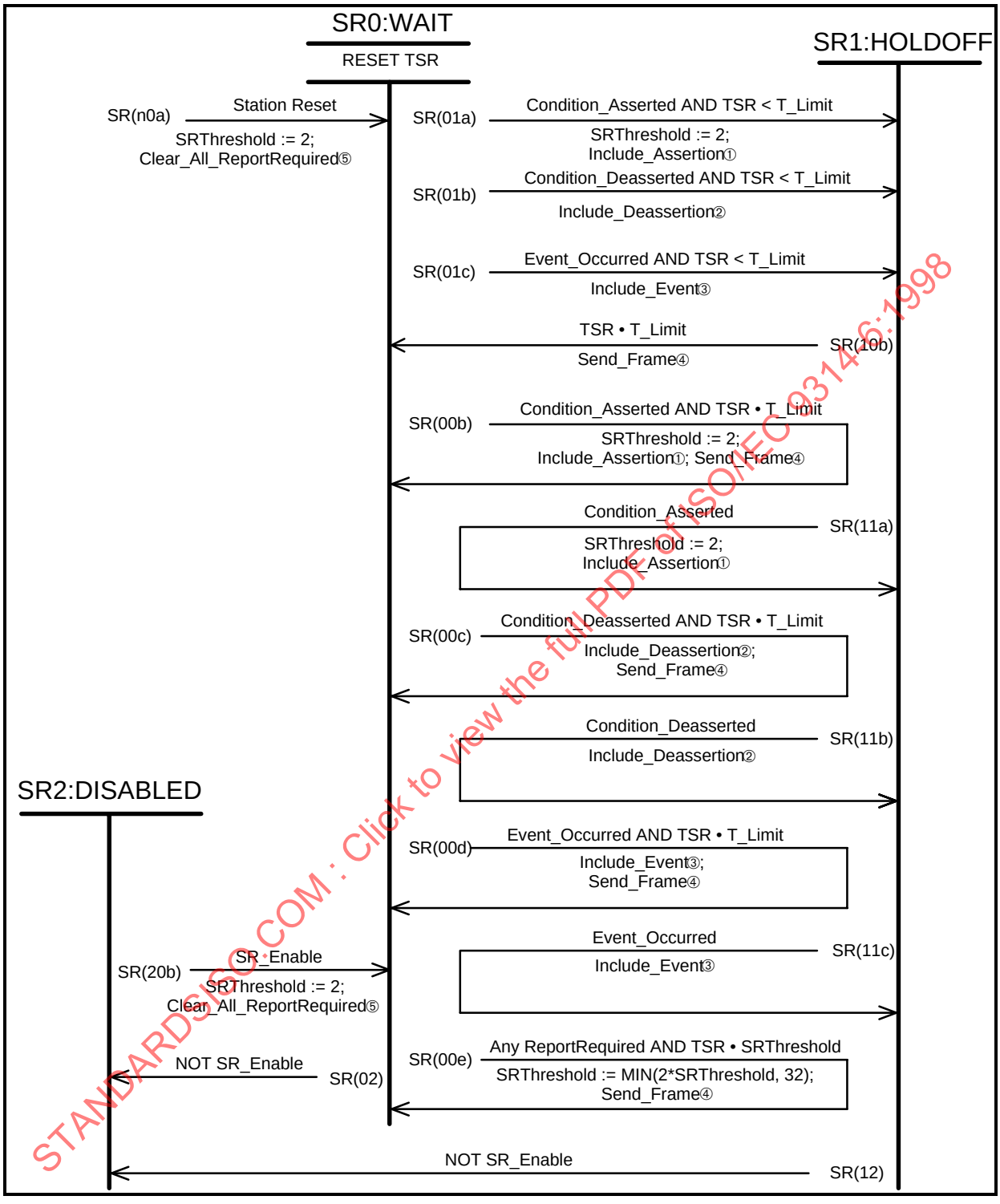


Figure 8 – Status Report transmitter state diagram (Part 1 of 2)

```
1. Include_Assertion:
      TransitionTimeStamp := TimeStamp;
      <object>.<condition>.ReportRequired := SET;
      <object>.<condition>.ConditionState := SET;
2. Include_Deassertion:
      TransitionTimeStamp := TimeStamp;
      <object>.<condition>.ConditionState := CLEAR;
3. Include_Event:
      TransitionTimeStamp := TimeStamp;
      IF <object>.<event>.ReportRequired = SET
            THEN
                  <object>.<event>.MultipleOccurrence := SET
            ELSE
                  <object>.<event>.ReportRequired := SET;
4. Send_Frame:
      RT_Flag := CLEAR;
      Transition ID := new Transaction ID; {  Successive values of transaction ID shall be
                                              distinct (not required across self-test or
                                              station initialization cycles). }
      Construct_Status_Report_Frame; {  Contains SMT_Parameters for all Conditions
                                        and Events with ReportRequired = SET (see 7.2.7) }
      FOR <All MACs> DO
            IF <MAC>.SM_MA_Avail
                  THEN
                        BEGIN
                              RT_Flag := SET;
                              queue(Status_Report_Frame)
                                    { via MAC_index:SM_MA_UNITDATA.request(asynchronous) }
                        END;
      IF RT_Flag = SET
            THEN
                  BEGIN
                        Clear_Event_ReportRequired⑥;
                        Clear_Deasserted_Condition_ReportRequired⑦
                  END;
5. Clear_All_ReportRequired:
      Clear_Event_ReportRequired⑥;
      FOR <All instances of all managed objects> DO
            FOR <All Condition types for object> DO
                  BEGIN
                        <object>.<condition>.ReportRequired := CLEAR;
                        <object>.<condition>.ConditionState:= CLEAR
                  END;
6.    Clear_Event_ReportRequired:
      FOR <All instances of all managed objects> DO
            FOR <All Event types for object> DO
                  BEGIN
                        <object>.<event>.ReportRequired := CLEAR;
                        <object>.<event>.MultipleOccurrence := CLEAR
                  END;
7.    Clear_Deasserted_Condition_ReportRequired:
      FOR <All instances of all managed objects> DO
            FOR <All Condition types for object> DO
                  IF <object>.<condition>.ConditionState = CLEAR
                        THEN
                              <object>.<condition>.ReportRequired := CLEAR;
```

Figure 8 – Status Report transmitter state diagram (Part 2 of 2)

8.3.3.3 State SR2:DISABLED

The status transmitter is in the disabled state when status reporting has been disabled by clearing the SR_Enable flag.

SR(20b): SR_Enable: The status transmitter enters the wait state when status reporting has been enabled by setting the SR_Enable flag. SRThreshold is reset to its initial value, and all ReportRequireds are cleared.

8.4 Parameter Management protocol

8.4.1 Overview

Remote management of station attributes is accomplished via a class of SMT frames called Parameter Management Frames (see 7.2.8). The Parameter Management (PM) protocol operates on all SMT Management Information Base (MIB) attributes, attribute groups and actions. The encodings for all attributes, including the Get request and Set request forms, is provided by the MIB definition (see 6.4), in conjunction with the SMT encoding rules (see 7.1.4).

8.4.1.1 Consistency control

The Parameter Management protocol includes a consistency control mechanism that prevents one requester from changing an attribute or invoking an action with a PMF Set operation when there has been an intervening change or action by another requester. Use of this consistency control mechanism for a Set operation is the choice of the requester.

This consistency control mechanism is implemented with a SetCount maintained by each station. The SetCount changes whenever an attribute is changed or an action is invoked via the SMT frame-based management protocols or by a management agent process through the SMT local management interface. The SetCount is carried in all PMF response frames (unless the PMF Set protocol is not implemented), and in the Station Information (operation and configuration) response frames.

The SetCount consists of a count which increments each time an attribute is changed by either the local or remote methods, along with the value of fddiSMTTimeStamp at the time of the change. The SetCount is returned to the requester in an initial response and is used by the requester in the subsequent Set request. The responder then checks the SetCount in the Set request to ensure an exact match with the current SetCount before permitting the requested operation.

8.4.1.2 Access control

The Parameter Management protocol provides support for an access control mechanism for Set operations. An Authorization parameter (see 7.3.1.30) may be included in Set requests. The length and contents of this parameter are defined by the implementer.

A target station that implements a PMF access control mechanism may grant or deny access based on any criteria, including the presence, absence, or contents of the authorization parameter; the presence or absence of a SetCount parameter; the source station ID; the requested operation; the target attribute; or any combination of these or any other criteria.

Target stations that do not implement an access control mechanism shall permit (subject to other PMF protocol requirements) all Set operations, independent of the presence or absence of an Authorization parameter in the request frame.

8.4.2 Operation

A requester may issue a PMF request at any time and to any destination, for the purpose of accessing the FDDI Management Information Base of one or more remote stations.

The PMF protocol provides two operations: Get and Set. The protocol is broken into two descriptions, one for the requester and one for the responder.

8.4.2.1. Requester

8.4.2.1.1 Get request

Get requests are used to retrieve the values of MIB attributes. To perform a Get operation:

a) Construct a PMF Get request frame (see 7.2.8.1) containing SMT_Parameters specifying the MIB attributes and/or attribute groups to be retrieved and transmit the frame to the target destination. Multiple attributes or groups may be requested for specific instances of MIB objects in a single PMF Get request frame.

The value of an attribute or group may be requested for all instances of an object type by specifying a resource index of zero for that attribute or group. Multiple requests of this type may be specified in a single PMF Get request, however, the frame shall not include both SMT_Parameters with resource index of zero and SMT_Parameters with resource index other than zero.

b) Upon receipt of the corresponding response frame (a PMF Get response containing a Transaction ID matching that of the request) the requester may examine the reason code parameter to determine the success or failure of the requested operation.

1) If the reason code is 'Success', the response frame includes SMT_Parameters containing the attribute values and the timestamp, and may include the SetCount and the LastSetStationID. The requester may use the SetCount in the response frame to attempt a subsequent Set operation on the target station. Because of potential changes to attributes while the response frame is being constructed, attributes in a response frame may in some cases appear to be inconsistent with one another.

2) All other reason codes indicate that the target station was unable to completely carry out the requested Get operation. In this case, the response includes some or none of the attribute values, the timestamp, and may include the SetCount and the LastSetStationID.

8.4.2.1.2 Set request

Set requests are used to modify the values of MIB attributes or invoke MIB actions. To perform a Set operation:

a) Construct and transmit a PMF Set request frame (see 7.2.8.3) containing an SMT_Parameter specifying the MIB attribute to be modified or the MIB action to be invoked.

The request optionally includes a SetCount SMT_Parameter containing a value previously obtained from the target station. The presence of the SetCount requires that the receiving station check the SetCount against its current SetCount value, if the SetCount is not present, the operation shall be performed without this check.

The request may also include an Authorization parameter.

b) Upon receipt of the corresponding response frame (a PMF Set response or request Denied Frame containing a Transaction ID matching that of the request) the requester may examine the reason code to determine the success or failure of the requested operation.

1) If the reason code is 'Success', the response frame includes SMT_Parameters containing the modified attribute value (unless the request specified an action), the updated SetCount, the timestamp, and the LastSetStationID (in this case, the StationID of the requester). The requester may use the SetCount in the response to attempt another Set request.

2) If the reason code indicates 'BadSetCount', the response frame includes parameters containing the unmodified attribute value (unless the request specified an action), the SetCount, the timestamp, and the LastSetStation ID. The requester may retry its Set request with the SetCount contained in the response.

3) All other reason codes indicate that the target station was unable to carry out the requested operation. The response frame includes parameters containing the SetCount, the timestamp, and the station ID of the last station to modify the MIB on the target station. A parameter containing the attribute value may be included in the response.

8.4.2.2. Responder

8.4.2.2.1 Get response

On receipt of a PMF Get request that satisfies the common SMT protocol requirements (see 8.1.3), the responder will process the request subject to the exception conditions enumerated below. If any of these conditions are satisfied, the recourse specified is performed, and processing of the request frame is terminated. When multiple attributes are requested and an exception condition is encountered, the response may contain a subset of the requested values. If more than one of the exception conditions are simultaneously satisfied, then the implementer may choose which of the recourses is performed.

a) The responder shall construct a PMF Get response frame containing reason code = 'No Parameter' and transmit it to the requester when any SMT_Parameter in the request frame

   1) specifies an attribute that is not supported or not recognized,

   2) specifies a Resource_Index of zero and none of the objects in the class support the attribute,

   3) specifies an attribute group and none of the attributes in the group are supported.

b) If the request specifies an attribute that is not readable through the MIB, for example, an action or a parameter from the X'00 xx' set, the responder shall construct a PMF Get response frame containing reason code = 'IllegalOperation' and transmit it to the requester.

c) If the Get request contains a parameter length that is not valid for the requested parameter type, the responder may optionally construct a PMF Get response frame containing reason code = 'LengthError'' and transmit it to the requester.

d) If the resultant response frame would exceed the maximum SMT frame length (see 7.1.2.5), the responder shall construct a PMF Get response frame containing reason code ='FrameTooLong' and transmit it to the requester. Optionally, the responder may include partial response data in that response frame, up to the maximum permitted frame length (see 7.1.2.5).

In the absence of any of these exception conditions, the responder shall construct and transmit to the requester a PMF Get response frame containing reason code = 'Success' and SMT_Parameter(s) containing the current value(s) of the requested attribute(s), or in the case where an attribute group was requested, the current value for each supported attribute within the requested group.

The current SetCount and LastSetStationId is also included if the Set protocol is supported. The SetCount and LastSetStationId shall be sampled prior to gathering the attribute values for the response.

8.4.2.2.2 Set response

On receipt of a PMF Set request that satisfies the common SMT protocol requirements (see 8.1.3), the responder will process the request subject to the exception conditions enumerated below. If any of these conditions are satisfied, the recourse specified is performed, and processing of the request frame is terminated. If more than one of the exception conditions are simultaneously satisfied, then the implementer may choose which of the actions is performed.

a) If the request specifies an attribute that is not supported or not recognized by the responder, the responder shall construct a PMF Set response frame containing reason code = 'No Parameter' and transmit it to the requester.

b) If a Set request specifies an attribute that is not writeable through the MIB (a read-only attribute, a parameter from the X'00 xx' set, or an attribute group), the responder shall construct a PMF Set response frame, containing reason code = 'IllegalOperation' and transmit it to the requester.

c) If a Set request specifies a target value that is not within the bounds specified for the target attribute, the responder shall construct a PMF Set response frame containing reason code = 'OutOfRange' and transmit it to the requester. Note that the bounds for an attribute may be defined directly in the MIB, or indirectly by the values of other attributes or groups of attributes in the MIB (see 6.4).

d) The responder may reject a Set request if it determines that the requester is not authorized to perform the Set operation. In this case, the Responder shall construct a PMF Set response frame containing reason code = 'NotAuthorized'.

e) If the Set request contains a parameter length that is not valid for the requested parameter type, the responder may optionally construct a PMF Set response frame containing reason code = 'LengthError'' and transmit it to the requester.

f) If the resultant response frame would exceed the maximum SMT frame length (see 7.1.2.5), the responder shall construct a PMF Set response frame containing reason code = 'Frame Too Long' and transmit it to the requester.

g) If the request frame contains a SetCount and it does not exactly match the current value of SetCount maintained by the responder, the responder shall construct a PMF Set response frame containing reason code = 'BadSetCount', the current value of the target attribute, and the current value of SetCount. The response shall be transmitted to the requester.

If none of the above exception conditions have occurred, the responder shall carry out the requested operation in the manner described below.

a) The new value of SetCount shall be calculated and stored. This value is calculated by incrementing the count and storing the current value of fddiSMTTimeStamp (see 6.4.5).

b) The responder shall record the StationID of station performing the Set operation in fddiSMTLastSetStationid (see 6.4.5).

c) Modify the target attribute, or invoke the action, as specified in the Set request. Any change to target attribute shall directly affect the active value of that attribute and the new value shall take effect before the response frame is constructed and transmitted.

d) A PMF Set response, containing a reason code = 'Success', the revised value of SetCount, and the revised value of the modified parameter shall be constructed and transmitted to the requester.

8.5 Station Status polling

8.5.1 Overview

A mechanism is provided for aggregate Station Status (SS) to be obtained remotely through a polling (request/response) protocol. This protocol is carried out using the SMT Status Information Frames (see 7.2.2). Two classes of Status Information Frames are defined: SIF Configuration, which carries several station connection and configuration parameters, and SIF operation, which carries some station statistical information.

8.5.2 Operation

A station may issue SIF request frames at any time and to any destination.

The recipient of a SIF request is required to respond with a SIF response of the same class unless the response would exceed the maximum SIF length specified in 7.1.2.5, in which case a request Denied frame with reason code = 'Frame Too Long' shall be sent in response.

Some of the parameters that are included in the SIFs are derived from the state of the MAC that transmits the SIF. The fddiMACSMTAddress of this MAC appears as the Source Address of the SIF.

8.6 Echo protocol

8.6.1 Overview

The SMT Echo protocol is provided for SMT-to-SMT loopback testing on an FDDI ring. The Echo protocol is carried out using SMT ECHO frames (see 7.2.3.), which may contain any amount up to the maximum frame size supported by ISO/IEC 9314-1 and ISO/IEC 9314-2 (see 7.1.2.5), of implementation-specific Echo data.

The length of the Echo data is constrained only by the maximum frame size supported by ISO/IEC 9314-2. Accounting for MAC, SMT, and SMT_Parameter header, the maximum data length for an Echo request is 4 454 octets.

MaxEchoData = MaxSMTInfo - SMT_ParameterHeader
= 4 458 - 4
= 4 454

8.6.2 Operation

The ability of a station to generate ECHO request frames is not required. A station may issue an ECHO request at any time, and to any destination.

The recipient of an ECHO request frame is required to respond with an ECHO response frame, unless the request exceeds the maximum ECF length specified (see 8.6.1), in which case the request shall not be processed (see 8.1). This ECHO response frame is constructed with the SMT_Info field from the received ECHO request.

8.7 Synchronous Bandwidth Allocation

8.7.1 Overview

This subclause describes the Synchronous Bandwidth Allocation (SBA) protocol employed on an operational FDDI logical ring. The protocol provides for a deterministic mechanism for allocation of synchronous bandwidth and supports monitoring for the over-allocation of synchronous and total bandwidth.

The SBA protocol supports the following functions:

a) manage the allocation of the limited synchronous bandwidth resource;

b) monitor the amount of synchronous bandwidth allocated for use;

c) monitor the ring for over-allocation of synchronous bandwidth; and,

d) monitor for and recover from ring instability due to over-allocation of the total bandwidth.

The protocol performs the allocation functions through a request-response frame exchange between a station using synchronous bandwidth and a synchronous bandwidth management process. Monitoring is done by the management process requesting allocation information from stations using synchronous bandwidth. The frames used are the SMT Resource Allocation Frames (see 7.2.4).

The SBA protocol is used by each MAC in a station that issues synchronous frames.

The synchronous bandwidth management process is not defined as part of this protocol. The facilities and operations defined here are designed to support and provide flexibility for the allocation algorithm of the management process.

8.7.2 Operation

The ability of a station to generate Resource Allocation Frames is not required unless the station uses synchronous bandwidth.

The SBA protocol assumes one central allocator of the synchronous bandwidth resource. The protocol provides for the management of synchronous bandwidth, the detection of over-allocation, and ability to recover from faults due to over-allocation. Specifically, the protocol supports the handling of four specific conditions:

a) Allocation/deallocation: stations desiring the use of or relinquishing the allocation of synchronous bandwidth register their requests with the management process.

b) Over-allocation of the available synchronous bandwidth: this occurs when the amount of synchronous bandwidth allocated exceeds fddiPATHSbaAvailable. This generally occurs when joining two logical rings supporting synchronous service.

c) Over-allocation of the total bandwidth: when the total amount of synchronous bandwidth allocated exceeds the current T_Neg less the ring latency. Should the stations use the synchronous bandwidth to this capacity, the ring will become unstable and oscillate between claiming and operational. This generally occurs when joining two logical rings supporting synchronous service.

d) Change to T_Neg: a change to the negotiated TTRT changes the synchronous bandwidth utilization.

8.7.2.1 Allocation

A station must receive an allocation of synchronous bandwidth before initiating synchronous transmission. To support the characteristics of synchronous data flows, the allocation protocol has been designed to maximize continuous operation in a dynamic network. This is accomplished by granting allocations to stations in two portions: synchronous payload and synchronous overhead. This division allows stations to adapt to a change in T_Neg without reallocating bandwidth in most cases.

A station that wants to change its current allocation of synchronous bandwidth must request the new allocation amount from the synchronous bandwidth management process. This includes initially requesting an allocation, increasing or decreasing the currently allocated amount, and releasing all bandwidth allocated to the station.

The overhead allocation is determined by the size of fixed fields that are used to create and transmit a synchronous frame. This includes token capture (once per station using synchronous bandwidth), and MAC frame overhead (including preamble), and any higher layer protocol overhead that is carried with the data flow. For example, for the first synchronous allocation in a station using long addresses and a small higher layer protocol overhead, the allocation would be:

$$\text{token\_capture} + \text{frame\_overhead} + \text{hlp\_overhead} = \text{SBA\_Overhead}$$

$$11 + 28 + 8 = 47$$

The station must also determine the allocation for the synchronous payload being carried in the frame. This is determined by calculating the bytes per 125 µs required to service the data flow. Any fraction is rounded up. For example, 1 megabit per second of synchronous payload requires an increase in the payload allocation to 16.

$$(10^6 / 8) * 125 \times 10^{-6} = 15{,}625$$

The synchronous payload is segmented into packets based on the current T_Neg. For example the above 1 megabit per second payload would be segmented into 500-byte packets for a 4 ms T_Neg and 1 000 bytes for an 8 ms T_Neg. In the request, the station also indicates the longest T_Neg acceptable for the bandwidth being requested, as well as the minimum size to which the synchronous payload will be segmented. These attributes allow the management process to more accurately evaluate the impact of granting the requested bandwidth. The request also includes the allocation address and a category SMT_Parameter which may be used by the allocation management process to provide for sophisticated management algorithms.

To request a change in allocation, a station sends an RAF request (ref 7.2.4.1) with Destination Address=(SMT-SBA-DA), frame class=X'41' (SMT Info) and SBA command=X'00 00 00 01' (Request Allocation). A management process will respond with an RAF response (see 7.2.4.2) indicating success or failure of the request. If the request fails, the station may adjust its allocation request to within the available bandwidth specified in the response frame and restart the exchange.

The station changes its synchronous bandwidth attributes when the response indicates successful allocation; or when the request is for deallocation of bandwidth. The management process is responsible for monitoring bandwidth to recover bandwidth which is either not deallocated, or for which deallocation request frames are lost. If an RAF response is not received for an allocation, the station should periodically repeat the request using the same Transaction ID. The station may not increase its synchronous bandwidth allocation until an RAF response is received from the management process. The station may decrease its current allocation after sending at least one RAF request. Optionally, after decreasing its current allocation, the station may continue to periodically send requests until a response is received.

A station may receive an allocation of bandwidth which it in turn may suballocate to end stations. In this case, the suballocating station will decrease fddiPATHSbaPayload and fddiPATHSbaOverhead to reflect suballocations, and the end station receiving a suballocation will show a corresponding increase in these attributes. A deallocation from an end station to a suballocating station operates in the reverse with accounting for bandwidth moving from the end station to the suballocating station.

Static allocation of synchronous bandwidth is desirable in some networks. The allocation process for these systems is similar. A static allocation must properly include payload and overhead as described above. A station using a static allocation must also properly respond to requests for Change Allocation or Report Allocation commands

All stations using synchronous service shall set bit 6 in its station state SMT_Parameter (see 7.3.1.3) contained in transmitted NIF frames. This allows a management process to monitor the synchronous service.

8.7.2.2 Monitoring

To maintain accurate allocation information, the management process should periodically request a station to report its current allocation. The management station may do this by using an RAF request with SBA command=X'00 00 00 02' (Report Allocation) (see 7.2.4.1) to stations utilizing synchronous bandwidth. The management station may learn the stations using synchronous service by monitoring NIF frames, or by maintaining information from the allocation request frames (e.g. allocation address).

When a station receives an RAF request (Report Allocation), it generates an RAF response (see 7.2.4.2) with Destination Address=(SA from request frame) and SBA command=X'00 00 00 02' (Report Allocation), and includes the station's current allocation for non-local data Paths.

The management application can then use the response to validate its allocation database. This monitoring function allows recovery of synchronous bandwidth that is not properly deallocated, either from frame loss or a station leaving the network without deallocating bandwidth.

8.7.2.3 Claim processing

Each successful completion of the Claim Token process may result in a new T_Neg. To properly utilize its allocated synchronous bandwidth, a station must recalculate the segment size of its payload, i.e. the amount of data within each frame transmitted on each token opportunity, excluding the overhead. A new T_Neg value does not require adjusting either fddiPATHSbaPayload or fddiPATHSbaOverhead, but it does require the station to adjust the segment size for information included in synchronous frames.

A significantly large change in T_Neg may cause over-allocation of synchronous bandwidth. The management process recognizes this condition and recovers from it.

8.7.2.4 Recovery

When the synchronous bandwidth management process determines that an over-allocation of bandwidth exists, it may invoke a recovery process to maintain ring availability. In cases where the total amount of synchronous bandwidth allocated exceeds a maximum value (specified in the management process), the management process may change the allocated bandwidth for some or all stations currently allocated synchronous bandwidth. In extreme cases, when the total network bandwidth is exceeded and the ring becomes unstable, the management process will force all stations to stop using synchronous bandwidth.

To change the allocated amount of synchronous bandwidth, a management process sends an RAF request (see 7.2.4.1) with Destination Address=(broadcast, group, or individual) and SBA command=X'00 00 00 03' (Change Allocation). Upon receipt of this frame, the station changes its

allocation to a value equal to or less than the allocation specified in the frame. An RAF response (see 7.2.4.2) with Destination Address=(SA from request frame) and SBA command=X'00 00 00 03' (Change Allocation) is generated to acknowledge the deallocation.

When the ring becomes unstable due to overuse of the total bandwidth by synchronous frames, the management process is responsible for recovery. In order to recover from this situation, the management process must force enough stations to stop using synchronous bandwidth so that the ring will become stable. This can be accomplished by sending a Change Allocation RAF request to the broadcast address. To restore synchronous service, the station must request an allocation from the management process.

8.7.3 Synchronous bandwidth management process

This part of ISO/IEC 9314 does not specify the management process that performs synchronous bandwidth allocation, but certain features must exist in the allocation process to assure proper operation of a ring supporting synchronous transmission. This subclause addresses some of these features.

Management of the synchronous bandwidth allocation occurs through a request-response frame exchange between a station and the management process. The management process, upon receipt of a request for allocation, must grant or deny the request. The mechanism for making the determination is left to the implementation of the management process. However, it is recommended that such a mechanism include the following:

a) Select a maximum allocation amount appropriate to the network — ring reconfigurations can easily cause problems of over-utilization. A single fault can join two rings into a single ring. In environments where ring reconfigurations are common or expected, it is suggested that the total allocatable bandwidth for both the primary and secondary rings be less than the usable bandwidth. This can be done by splitting the allocation between the rings or placing all of the synchronous traffic on one of the rings. With this value, a wrap that joins the two rings with maximum allocation will not cause over-utilization of the total bandwidth. In stable environments, a larger total allocation may be used with care.

b) Do not allocate 100 % of the total usable bandwidth for synchronous transmission — the payload amount of Synchronous Bandwidth Allocation (fddiPATHSbaPayload) measures the allocation in 125 µs units. The amount of synchronous bandwidth allocated must be evaluated in terms of T_Neg. In some cases, T_Neg will not be an integral multiple of the synchronous bandwidth requested by a station. This may cause a station to over-utilize by a small margin. The fact that some stations may over-allocate in mapping the bandwidth required to 125 µs units mitigates this problem, but it does not guarantee under-utilization of the allocation.

During operation, the management process monitors the ring for over-allocation conditions. To accurately track the amount of synchronous bandwidth allocated, the management process polls the stations' allocated bandwidth. The station state SMT_Parameter (see 7.3.1.3) is included in all NIFs (see 7.2.1). As part of the Neighbour Notification protocol (see 8.2), NIFs will be sent to the broadcast address at a regular interval. The synchronous bandwidth management process may passively monitor these frames to build a list of stations that believe they have been allocated synchronous bandwidth. From this list, the management process can poll the stations to determine the total amount of synchronous bandwidth allocated for use. This polling process allows the management process to maintain an acceptably accurate accounting of allocated bandwidth while stations appear and disappear due to faults and ring configuration changes.

Once the management process has determined the total amount of allocated bandwidth, it may determine if an over-allocation situation exists. The two over-allocation conditions are over-allocation of the allowable synchronous bandwidth and over-allocation of the total bandwidth. The second condition includes the first condition and may be handled through the same mechanism as long as the ring is stable (i.e. the stations are not fully utilizing the allocated bandwidth).

The maximum synchronous bandwidth allocatable is specified by fddiPATHSbaAvailable. An allocator may, depending on the value of fddiPATHSbaAvailable, establish a high-water mark slightly greater as the trigger for recovery. This allows the network to recover through attrition rather than forcing stations to reduce their services. However, when the allocated bandwidth exceeds this high-water mark, the synchronous bandwidth management process would actively force stations to reduce their allocation.

Sending an RAF request (see 7.2.4.1) with an SBA command = X'00 00 00 03' (Change Allocation) to a station can force the station to reduce its bandwidth allocation.

Over-allocation may occur when T_Neg changes to a smaller value or when the ring configuration changes. In both cases, a Claim Token process will occur. Thus, after the completion of any claim process, the management process should check T_Neg, and if it changes, recalculate SBA Available to check for over-allocation. Repeated and periodic ring oscillation is a symptom of over-allocation or over-utilization of the total bandwidth. If this occurs, especially after a configuration change, the management process should determine the cause.

Because the SBA protocol uses a central allocator, it may be desirable to have backup allocators available to assure availability of the service. If more than one allocator exists, a protocol must be defined among the allocators to determine which station contains the active management process and which stations become stand-by allocators. The communication among allocators may utilize synchronous frames or employ other SMT frames. A robust SBA management process will periodically poll for the existence of another SBA allocator by sending a mandatory SMT request frame to the SMT-SBA-DA. This will cause a response from either a backup allocator or a station running a different allocation protocol.

8.8 Extended Service protocol OPTIONAL

The Extended Service (ES) protocol is defined for extending and exercising new SMT frame-based services. All Extended Service frames (see 7.2.6.) carry an ESF_ID SMT_Parameter that uniquely identifies (through the use of IEEE-assigned Organizationally Unique Identifiers) the extended service being supplied. The protocol and semantics of these frames is specific to the particular ESF_ID carried by the frames.

If the recipient of an Extended Service request does not support the particular service specified by the ESF_ID, it shall ignore the request.

9 Connection Management

9.1 Overview

The operation of Port insertion and removal, and the connection of the Port PHY entities to the Media Access Control (MAC) entities are Station Management functions. The portion of Station Management that performs these services is called Connection Management (CMT). The formal specifications of the operation of CMT are contained herein.

As described, the FDDI CMT supports a wide variety of physical and logical topologies. A physical attachment to FDDI comprises PHY and PMD entities (Ports) and MAC entities. Connection Management controls and directs the establishment of media attachment to the FDDI network, the physical connections with the Ports of other stations (nodes), and the internal configuration of the interconnections of the various MAC and PHY entities within stations, in a way that allows interoperability between stations and concentrators. Services provided by CMT include:

– initialization of physical connections
– control of the optional optical bypass switch
– connection continuity test
– support of connection policies
– support of availability policies
– local Loop with neighbour MAC
– placement of available MACs
– removal of orphan PDUs
– detection of faults
– reconfiguration around faults
– testing of link confidence
– monitoring of link quality
– withholding of unacceptable connections
– support of fault tracing function
– invocation of Path Test
– support of maintenance line states
– indication of connection availability

9.2 Organization

The Connection Management function is further subdivided into these areas:

a) The management of the media interface to the FDDI network, which includes coordinating the activity of all of the Ports associated with that physical attachment and controlling the optional optical bypass function within the node. This area is called Entity Coordination Management (ECM) (see 9.5). There is a single instance of Entity Coordination Management for an FDDI node.

b) The management of a physical connection between the Port being managed and another Port, likely in an adjacent station or concentrator, on the FDDI network. This area is called Physical Connection Management (PCM) (see 9.6). An instance of Physical Connection Management exists for each Port in an FDDI node.

c) The management of the configuration of the MAC and Port entities within a node. This area is called Configuration Management (CFM) (see 9.7). There is a single instance of CFM for each resource in an FDDI node.

9.3 Connection Management structure

The relationships between the three areas of Connection Management and the FDDI PMD, PHY and MAC layers are shown in figure 9. Figure 9 is for illustrative purposes and does not define or imply interfaces that have been defined elsewhere in this part of ISO/IEC 9314.

9.4 Facilities

The following variables, flags, signals, and timers are used within Connection Management for maintaining input parameters, as well as maintaining the current status of the state machines. The context of the line states for the signalling of information is also defined.

9.4.1 Variables

A variable shall take on one value from a limited set of possible values. When a variable is cleared, the value of the variable becomes "None." Variables may be exported outside of Connection Management.

Path_Test A variable from a management entity to ECM used to indicate the Path Test status. An FDDI node has a single instance of Path_Test. Valid values of Path_Test are:

- None - The Path_Test variable is set to None when no Path Test status exists.
- Testing - The Path_Test variable is set to Testing while a Path Test is being performed.
- Passed - The Path_Test variable is set to Passed when the Path Test is successfully completed.
- Failed - The Path_Test variable is set to Failed when the Path Test fails.
- Pending - The Path_Test variable is set to Pending to signify that the Trace function is complete and a Path Test will follow.
- Exiting - The Path_Test variable is set to Exiting when a Disconnect is issued in the Trace or Leave states.

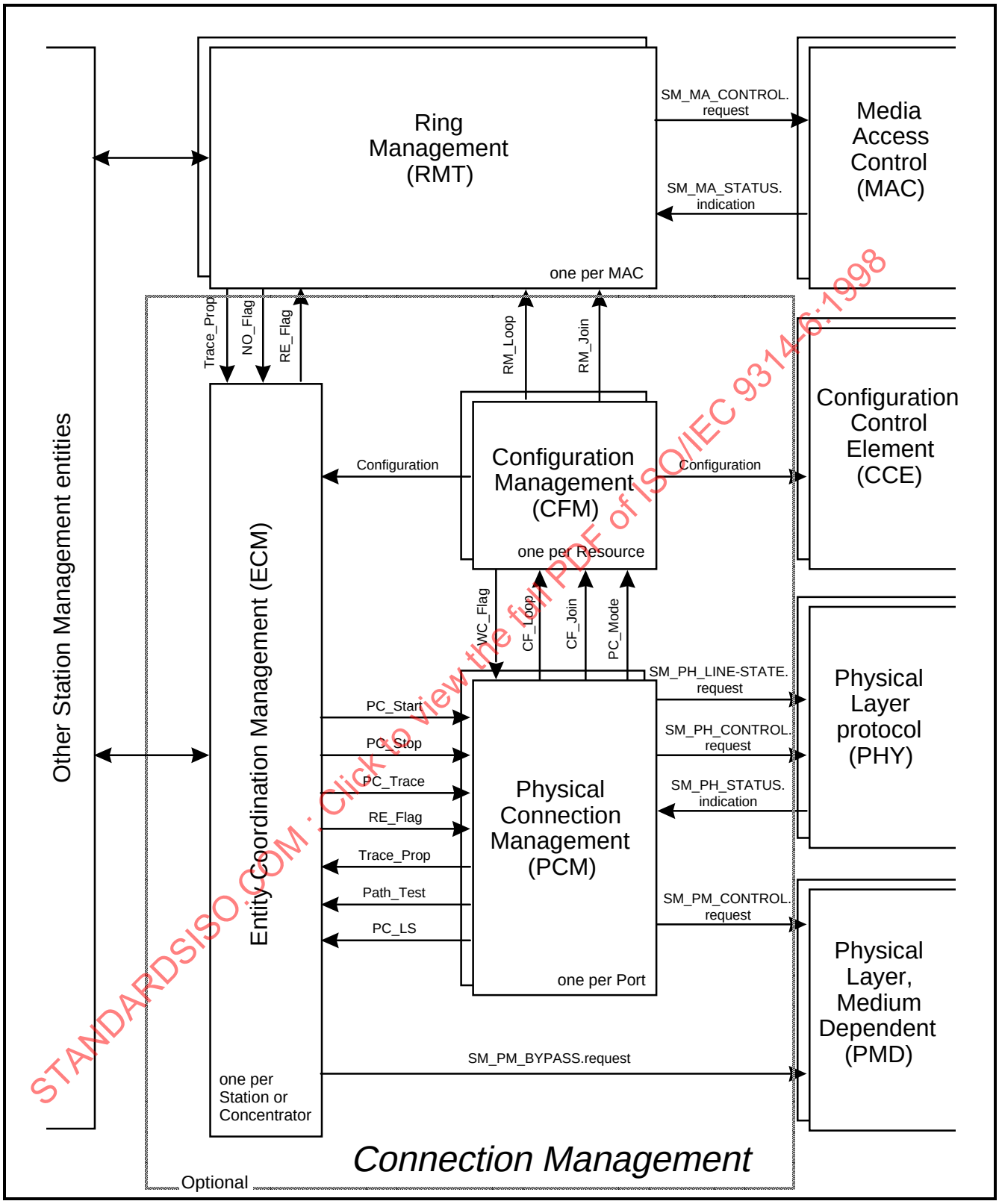


Figure 9 – Connection Management structure

PC_Type A variable that specifies the type of Port connector on the Port being managed by the PCM. Each Port connector type has a specific keying on its associated MIC receptacle. The four different Port connector types are:

A - The Port in a Dual Attachment Station or Dual Attachment Concentrator that attaches to the Primary In and the Secondary Out when attaching to the dual ring.

B - The Port in a Dual Attachment Station or Dual Attachment Concentrator that attaches to the Secondary In and the Primary Out when attaching to the dual ring.

S - One Port in a Single Attachment Station shall be designated S. One Port in a Single Attachment Concentrator shall also be designated as S.

M - A Port in a concentrator that serves as a Master to a connected node shall be designated M.

PC_Neighbour A variable from PCM to other management entities that specifies the type of Port Connector at the other end of the physical connection. The variable PC_Neighbour may take on a value of A, B, S, M, or None. The variable PC_Neighbour takes on the same value as PC_Type in the neighbouring Port, and is received during the PCM signalling.

PC_Mode A variable from PCM to other management entities indicating the mode of physical connection that was formed. The valid values of PC_Mode are:

Peer (P) - The PC_Mode is set to Peer when neither the Port under control nor the Port at the other end of the connection are of type M. This mode of connection exists within the trunk ring.

Tree (T) - The PC_Mode is set to Tree when a Port at one and only one end of the connection is of type M. This mode of connection exists within a concentrator tree.

None (N) - The PC_Mode is set to None when no other PC_Mode has been established.

PC_Withhold A variable from PCM to other management entities containing the reason for withholding a connection. Valid values of PC_Withhold are:

None

Port M to Port M

Other incompatible Port types

Requested Paths not available

Connect_State A variable from PCM to other management entities indicating the state of the connection. Values of Connect_State include:

Disabled

Connecting

Standby

Active

Maint_LS A variable from other management entities to PCM specifying the symbol stream to be transmitted when the PCM is in the Maint state. This variable is set to "None" by assigning a value of Quiet. The variable Maint_LS shall take on one of the following values:

TRANSMIT_QUIET

TRANSMIT_HALT

TRANSMIT_IDLE

TRANSMIT_MASTER

TRANSMIT_PDR

PC_LS A variable from PCM to other management entities providing the line state received by the PHY under control. The line state is determined using the SM_PH_STATUS.indication(status_report) primitive. This variable is set to "None" by assigning a value of LINE-STATE_UNKNOWN. The variable PC_LS shall take on one of the following values:

QUIET_LINE-STATE_RECEIVED (QLS)

HALT_LINE-STATE_RECEIVED (HLS)

MASTER_LINE-STATE_RECEIVED (MLS)

IDLE_LINE-STATE_RECEIVED (ILS)

ACTIVE_LINE-STATE_RECEIVED (ALS)

NOISE_LINE-STATE_RECEIVED (NLS)

LINE-STATE_UNKNOWN (LSU)

PC_LCT_Fail A variable from the Link Confidence Test (LCT) to other management entities containing the number of consecutive failures of the Link Confidence Test.

n An internal PCM variable containing the number of the next value to be signalled in the Next state and the current value being signalled in the Signal state.

9.4.2 Signals

A signal is used to initiate a state change within Connection Management. A signal causes the state change to occur and does not have to be cleared following its usage. A signal is local to Connection Management and may not be exported.

Connect A signal from a management entity to ECM indicating that the ECM state machine is to begin a connection sequence.

Disconnect A signal from a management entity to ECM indicating that the ECM state machine is to begin a disconnection sequence.

PC_Start A signal from a management entity to PCM indicating that the PCM state machine is to enter the Break state. PC_Start is signalled by ECM to start the PCM state machine and is signalled by PCM when the Link Confidence Test fails.

PC_Maint A signal from other management entities to PCM indicating that the PCM state machine is to enter the Maint state. PC_Maint causes a transition to Maint state only when the signal is issued while the PCM state machine is in the Off state.

PC_Trace A signal from a management entity to PCM indicating that the PCM state machine is to enter the Trace state.

PC_Stop A signal from a management entity to PCM indicating that the PCM state machine is to enter the Off state. PC_Stop causes a transition to the Off state from any state of the PCM except the Maint state.

PC_Enable A signal from a management entity to PCM indicating that the PCM state machine is to transition from the Maint state to the Off state.

PC_Disable A signal from a management entity to PCM indicating that the PCM state machine is to transition to the Maint state. PC_Disable causes a transition to the Maint state from any state of the PCM.

PC_Signal A signal internal to PCM indicating that the next value to be signalled is available and ready for transmission.

PC_PDR A signal internal to PCM indicating that a Transmit_PDR is desired on the Port to pass PDUs.

PC_Join A signal internal to PCM indicating that the PCM join sequence shall be started. The successful completion of the PCM join sequence leads to the generation of CF_Join, which indicates that the connection is ready to be incorporated into the token Path.

9.4.3 Flags

A flag is a variable that shall take on one of two values: set or cleared. Flags are assigned values within the state machines by specifying a set or clear operation. Flags are tested within state machines by checking their status or the negation of their status. Transitions initiated by flags need only be checked upon initial entry into the state and upon change of the flag. The value of a flag may be exported outside of Connection Management.

9.4.3.1 Operational flags

LS_Flag Line State flag. A PCM flag used to indicate that a given line state has been received since entering the current state.

RC_Flag Receive Code flag. A private PCM flag used to indicate that the receive pseudo code has started execution.

TC_Flag Transmit Code flag. A private PCM flag used to indicate that the transmit pseudo code has started execution.

TD_Flag Transmit Delay flag. A private PCM flag used to indicate that the execution of the transmit pseudo code is to be delayed.

T_Val(n) Transmitted value n. A single bit of information communicated from a PCM to a neighbouring PCM while in the Signal state.

R_Val(n) Received value n. A single bit of information containing the value received from the opposing neighbour Port for T_Val(n).

CF_Loop A flag from PCM to CFM indicating that the Port is ready to be incorporated into a local token Path.

CF_Join A flag from PCM to CFM indicating that the Port is ready to be incorporated into a non-local token Path.

RM_Loop A flag from CFM to RMT indicating that the MAC has been incorporated into a local token Path.

RM_Join A flag from CFM to RMT indicating that the MAC has been incorporated into a non-local token Path.

TR_Flag Trace Received flag. A flag internal to PCM that a Trace has been received while in the Active state.

Trace_Prop Trace Propagate flag. A flag from PCM or RMT to ECM indicating that a Trace is to be propagated internally.

RE_Flag Recovery Enable flag. A flag from ECM indicating that recovery is enabled. This flag is used in conjunction with the optional Hold Policy. For nodes that do not implement the Hold Policy, this flag shall be set.

WC_Flag Withhold Connection flag. A flag from CFM to PCM used to indicate that a Port connection is to be withheld. This flag shall be set when the connection is to be withheld as a result of evaluating the Requested Paths selection criteria. When there is a change in state of the WC_Flag, PCM shall reevaluate the exit conditions of its current state.

CF_Thru A flag internal to Configuration Management used to indicate that the Port under control shall be connected in a thru configuration when the Port becomes active. An instance of this flag exists for each A and B Port in a node. This flag shall be set as a result of evaluating the requested Paths selection criteria.

CF_Wrap A flag internal to Configuration Management used to indicate that the Port under control shall be connected in a wrap configuration when the Port becomes active. An instance of this flag exists for each A, B, and S Port in a node. For an A Port, the Secondary Path is wrapped to the Port. For a B or S Port, the Primary Path is wrapped to the Port. This flag shall be set as a result of evaluating the requested Paths selection criteria.

CF_C_Wrap A flag internal to Configuration Management used to indicate that the Port under control shall be connected in a concatenated wrap configuration when the Port becomes active. An instance of this flag exists for each A, B, and S Port in a node. This flag shall be set as a result of evaluating the requested Paths selection criteria.

CF_Insert_P A flag internal to Configuration Management used to indicate that the M Port under control shall be connected into the Primary Path when the Port becomes active. An instance of this flag exists for each M Port in a concentrator. This flag shall be set as a result of evaluating the requested Paths selection criteria.

CF_Insert_S A flag internal to Configuration Management used to indicate that the M Port under control shall be connected into the Secondary Path when the Port becomes active. An instance of this flag exists for each M Port in a concentrator. This flag shall be set as a result of evaluating the requested Paths selection criteria.

Attach_P A flag internal to Configuration Management used to indicate that the MAC under control shall be connected into the Primary Path. An instance of this flag exists for each MAC in a node. This flag shall be set as a result of evaluating the requested Paths selection criteria.

Attach_S A flag internal to Configuration Management used to indicate that the MAC under control shall be connected into the Secondary Path. An instance of this flag exists for each MAC in a node. This flag shall be set as a result of evaluating the requested Paths selection criteria.

Attach_L A flag internal to Configuration Management used to indicate that the MAC under control shall be connected into the Local Path. An instance of this flag exists for each MAC in a node. This flag shall be set as a result of evaluating the requested Paths selection criteria.

NO_Flag NonOperational flag. A flag used to indicate that the ring on which the MAC resides has been non-operational for an extended period.

TraceInitiated The trace initiated flag indicates that a trace has been initiated by a MAC on a Path in this node. An instance of this flag exists for each Path in the node.

TracePropagated The trace propagated flag indicates that a trace has been propagated by a Port on a Path in this node. An instance of this flag exists for each Path in the node.

TraceTerminated The trace terminated flag indicates that a trace has been terminated by a MAC on a Path in this node. An instance of this flag exists for each Path in the node.

TraceTimeOut The trace timeout flag indicates that a trace initiated or propagated on a Path in this node was not acknowledged. An instance of this flag exists for each Path in the node.

9.4.3.2 Capability flags

Bypass A flag internal to ECM that is used to indicate the existence of an optical bypass switch. Bypass is set for a CMT with an optical bypass switch and cleared otherwise.

Hold An optional ECM flag used to enable the optional Hold Policy. This policy flag may only be used in dual attachment dual MAC nodes. The flag is set if the Hold Policy is enabled (fddiSMTConfigPolicy) and the node supports Hold (fddiSMTConfigCapabilities).

PC_MAC_LCT A PCM flag that indicates that a MAC shall be used for the Link Confidence Test. This flag shall be set equal to the AND of the corresponding bits in the fddiPORTConnectionCapabilities and the fddiPORTConnectionPolicies attributes.

PC_MAC_Loop A PCM flag that indicates that MAC Local Loop shall be performed before the connection is made Active. This flag shall be set equal to the AND of the corresponding bits in the fddiPORTConnect-ionCapabilites and the fddiPORTConnectionPolicies attributes.

CF_MAC A flag from CFM to PCM to indicate that a MAC is available for either the Link Confidence Test or MAC local loop or both. The setting of CF_MAC is implementation dependent and varies depending upon the desired initialization sequence and availability policies.

NOTE 11 – Discretion should be exercised when using the MAC in a dual attachment single MAC station for the Link Confidence Test or MAC local loop as this may produce MACless wrap points on the dual ring.

9.4.3.3 Error indicator flags

SB_Flag Stuck Bypass flag. An ECM flag used to indicate that the ECM state machine is not proceeding and a stuck bypass switch is suspected. This flag is only present in nodes with optical bypass switches.

BS_Flag Break State flag. A PCM flag used to indicate that the PCM state machine is not leaving the Break state in an expected time interval and a problem is suspected.

LEM_Fail Link Error Monitor failure flag. A PCM flag that is set when the Link Error Rate (LER) exceeds the the LER_Cutoff threshold and cleared when the Link Error Rate threshold test is passed. The LEM_Fail flag may optionally be cleared in the PC_Connect state if TPC is significantly greater than T_Out. The LEM_Fail flag is used to remove connections with excessive link error rates and to establish the Link Confidence Test length.

9.4.4 Timers

9.4.4.1 Timer functions

TEC Timer, Entity Coordination. A timer used by ECM to allow sufficient time for insert, deinsert, and break.

TPC Timer, Physical Connection. A timer used by PCM for ensuring state transitions proceed at the desired rate.

TID Timer, Idle Detection. A timer used by PCM to measure the time of continuous ILS reception.

TNE Timer, Noise Event. A timer used by PCM to detect the length of noise events. This timer records the time since the last Idle Line State, Halt Line State, Quiet Line State, or Master Line State was received. Active Line State shall not reset the TNE timer because JK symbol pairs may occur frequently in random noise.

NOTE 12 – Optionally, the TNE timer may record the length of time from entering Noise Line State to receiving Idle Line State, Halt Line State, Quiet Line State, or Master Line State. Whereas this optional implementation will detect and react to noise events, it will not detect a jabbering MAC.

9.4.4.2 Timer expiration values

The specification of the expiration values for the Connection Management timers follows. The given range specifies inequalities necessary for interoperability. In the range specifications, the parameters are grouped into ring parameters, my node parameters, and all nodes parameters. Ring parameters are those that establish limits on the extent of the network, whereas my parameters are those that apply to the node under control, and all nodes parameters are those that may represent instances of the parameter in any node in the network. Inequalities involving all nodes parameters shall be satisfied by all instances of the parameter in the network. Using ring parameter values larger than the defaults to calculate the expiration values in a limited set of nodes may result in inconsistent recovery parameters and the failure of the network to recover from certain faults.

9.4.4.2.1 Elements of timer calculation

ANS_Max Maximum Signal_Detect deassertion time for a node when the optical input decays with a ramp function.

Range: ANS_Max ≤ 350 µs

Default: 350 µs

D_Max Maximum Ring Latency. This is the maximum circulation delay for a Starting Delimiter around the logical ring in the absence of noise. The default value for D_Max is specified in the PHY document.

Default: 1,773 ms

P_Max Maximum number of Physical Layer Entities. The default value of P_Max is specified in the PHY document. The range of P_Max is constrained by the selection of D_Max and Trace_Max.

Default: 1 000

LS_Max Maximum time to reestablish the correct line state as specified in the PHY document. SMT uses a larger default value.

Range: LS_Max ≤ 25 µs

Default: 25 µs

MI_Max Maximum optical bypass media interruption time. The range and default value for MI_Max is specified in the PMD document.

Range: MI_Max ≤ 15,0 ms

Default: 15 ms

PC_React Maximum time for PCM to make a state transition to Break upon reception of QLS.

Range: PC_React ≤ 3,0 ms

Default: 3 ms

CF_React Maximum time for the CFM to reconfigure to remove a non-Active connection from the token Path.

Range: CF_React ≤ 3,0 ms

Default: 3 ms

Trace_React Maximum time for an FDDI node to react to a Trace and propagate it on an outgoing link. The maximum time from when a Trace is received on an input to when the outputs of an affected PMD in the node changes to alternating Halt and Quiet symbols shall not exceed Trace_React.

Range: Trace_React ≤ 12 ms

Default: 12 ms

Scrub_Max Maximum Scrub time. Maximum time that the ring continuity is broken to remove old PDUs from the ring. A maximum limit is placed on scrubbing time so that the overall initialization time is bounded. The default value of Scrub_Max is less than or equal to 4(D_Max) or 2(TVX_Value). An upper limit of 2(TVX_Value) allows a scrubbing PHY to initiate the Claim process by causing the expiration of TVX in a MAC on the ring.

Range: Scrub_Max ≤ (4 × D_Max)
Scrub_Max ≤ 2 × (TVX_Value)

Scrub_Max ≤ 7.1 ms with default values

Default: 7,1 ms

9.4.4.2.2 ECM expiration values

I_Max Maximum optical bypass insertion/deinsertion time for this station. The default value for I_Max is specified in the PMD document.

Range: I_Max ≤ 25,0 ms

Default: 25 ms

IN_Max Maximum time after an insertion request until a response from an active neighbour may be observed on the inbound link. The total link Path may be as long as four links and four Ports for the fault condition of an optical bypass switch that inserted on only one of the dual rings. IN_Max is greater than the sum of:

I_Max	The maximum optical bypass insertion time for this station. The default value for I_Max is 25,0 ms.
D_Max	The maximum propagation delay of four physical links is bound by D_Max. The default value for D_Max is 1,773 ms.
3(LS_Max)	The maximum time required for three PHYs to recognize the line state.
3(PC_React)	The time for the other PCMs to transition to the Break state if a Trace is not present.
ANS_Max	The maximum time for QLS to be indicated after optical bypass switching.

Range: IN_Max ≥ ring(D_Max) +
my(I_Max + LS_Max + PC_React) +
all(2× LS_Max + ANS_Max + 2× PC_React)

IN_Max ≥ 36,198 ms with default values

Default: 40 ms

TD_Min Minimum time to transmit QLS before deinsertion. TD_Min shall be greater than the sum of:

LS_Max	Maximum time for the receiving PHY to the recognize the line state.
PC_React	Time for the PCM in the neighbouring Port to enter the Break state.

Range: TD_Min ≥ all(LS_Max + PC_React)

TD_Min ≥ 3,025 ms with default values

Default: 5 ms

Trace_Max Maximum propagation time for a Trace on an FDDI topology. Trace_Max places a lower bound on the detection time for a non-recovering ring (T_Stuck). Trace_Max is greater than the sum of:

D_Max — The maximum propagation delay for an entire FDDI cable plant. The default value for D_Max is specified as 1,773 ms.

P_Max(Trace_React) — The time for the Trace to propagate through all nodes on the FDDI.

Range: Trace_Max ≥ ring(P_Max/2) × all(Trace_React) + ring(D_Max)

Trace_Max ≥ 6,001 773 s with default values

Default: 7 s

9.4.4.2.3 PCM expiration values

C_Min Minimum time required to remain in the Connect state to assure that other end has recognized Halt Line State. C_Min is greater than the sum of:

TE_Max — This allows sufficient time for turn on of the single mode PMD. TE_Max is defined in the SM PMD document to have a maximum value of 1,0 ms.

A_Max — A_Max for the multimode connection. This allows sufficient time for the receiving PHY to achieve signal acquisition.

Range: C_Min ≥ all(TE_Max + A_Max)

C_Min ≥ 1,2 ms with default values

Default: 1,6 ms

TL_Min Minimum time to transmit a PHY line state before advancing to the next PCM state. TL_Min is set to twice the time required for line state recognition (LS_Max). This default value of TL_Min is specified for station and concentrator interoperability and may additionally constrain the implementation of FDDI repeaters.

Range: TL_Min ≥ all(2 × LS_Max)

TL_Min ≥ 50 µs with default values

Default: 50 µs

LS_Min Length of time continuous reception of ILS is required to be used by PCM. LS_Min shall be greater than or equal to 12 symbol times to ensure robustness, and is limited by LS_Max to ensure recognition when TL_Min time of a line state is received.

Range: all(TL_Min) - my(LS_Max) ≥ LS_Min ≥ 12 × Symbol_Time

25 µs ≥ LS_Min ≥ 0,48 µs with default values

Default: 0,48 µs

TB_Min Minimum Break time for link. TB_Min shall be sufficiently large to see a response on the inbound link. A link Path in the absence of faults contains two links and two PHYs. TB_Min shall be greater than the sum of:

D_Max — The maximum propagation delay of two physical links.

2(LS_Max) — The maximum time required for both PHYs to recognize the line state.

PC_React The time for the other PCM to transition to the Break state if a Trace is not present.

Range: TB_Min ≥ ring(D_Max) + my(LS_Max) + all(PC_React + LS_Max)

TB_Min ≥ 4,823 ms with default values

Default: 5 ms

TB_Max Break time before the BS_Flag is set. TB_Max shall be sufficiently large so that it will not be set inadvertently by noise generated by an optical bypass switch, which is bounded by MI_Max. A safety factor of n=2 is used.

Range: TB_Max ≥ all(n × MI_Max)

TB_Max ≥ 30,0 ms with default values

Default: 50 ms

T_Out Signalling timeout. The minimum time that a PCM state machine will remain in a state awaiting a line state change. When a line state change is expected and no transition is made in T_Out time, a transition shall be made to the Break state. This specification of T_Out places limits on the maximum time for making other transitions with PCM.

Range: T_Out ≥ 100 ms

Default: 100 ms

NS_Max The maximum length of time that noise, as measured by TNE, is allowed before a connection is broken down and restarted. The default value for NS_Max can be derived conveniently from the symbol clock, i.e. with a 15-bit counter. The length of time for NS_Max shall be:

a) Greater than the maximum length of two frames including a maximum preamble.This allows for errors to span a preamble, and ensures that the connection will not be broken down during normal frame transmission. NS_Max>0,725 5 ms to satisfy this criterion.

b) Less than the interval with probability of 0,5 of decoding Idle Line State. The shortest pattern that can be decoded as Idle Line State is 20 consecutive 1 bits. For uniform random noise, each of these bits has a P(1) = 0,5, so the probability of 20 such bits is $(0,5^{20})$. The number of bits *N* at which there is a 0,5 probability of observing four contiguous Idle symbols in random noise can be derived from:

$$0,5 = (1 - 0,5^{20})^{N}$$

Solving for *N* yields 726 820 bits.

(1 false detection /726 820 bits) x (125 x 10^6 bits/sec.) = 1 false detection per 5,8 ms

Thus, NS_Max < 5,8 ms for uniform random noise, and could be shorter for other noise models.

Range: 5,8 ms ≥ NS_Max ≥ 0,725 5 ms

Default: 1,3 ms

LC_Short Short Link Confidence Test time.

Range: LC_Short ≥ 50 µs

Default: 50 ms

LC_Medium Medium Link Confidence Test time.

Range: LC_Medium ≥ 500 ms

Default: 500 ms

LC_Long Long Link Confidence Test time.

Range: LC_Long ≥ 5 s

Default: 5 s

LC_Extended Extended Link Confidence Test Time. There is no upper limit on the value of LC_Extended for a particular implementation.

Range: LC_Extended ≥ 50 s

Default: 50 s

T_Next(7) LC_Test Time for Link Confidence Test.

T_Next(9) Time for the optional MAC Local Loop to prevent deadlock. This allows sufficient time for MAC recovery process completion and the exchange of neighbour information frames.

Range: T_Next(9) ≥ 200 ms

Default: 200 ms

9.4.4.2.4 CFM expiration value

T_Scrub Scrub Time. Time that the ring continuity is broken to remove old PDUs from the ring. T_Scrub is greater than the ring latency (D_Max) to accomplish this. T_Scrub is less than Scrub_Max. T_Scrub is bounded above to place a limit on initialization time.

Range: (Scrub_Max) ≥ T_Scrub ≥ (D_Max)

7,092 ms ≥ T_Scrub ≥ 1,773 ms with default values

Default: 3,5 ms

9.4.5 Line states

The bit synchronized signalling technique uses the line states supported by the PHY entity. The four PHY line states available for PCM signalling are:

Quiet Line State (QLS)
Halt Line State (HLS)
Master Line State (MLS)
Idle Line State (ILS)

The utilization of these line states for signalling makes use of their detection requirements as specified in ISO 9314-1. The line states with the strictest requirements for recognition are HLS and MLS. These two line states are used in the communication of signalled information.

The ILS provides the best maintenance of clock synchronization and is used to separate the communication of information. While ILS requires only four symbols for recognition by PHY, the recognition requirements of ILS for the PCM are more stringent. For the PCM to recognize ILS, the reception of 16 consecutive Idle symbols is required. Since four Idle symbols are required to initiate the recognition of ILS, the holding of ILS for 12 consecutive symbol times (LS_Min) ensures that 16 consecutive Idle symbols were received.

Since the reception of QLS from another PHY is not distinguishable from a disconnected connector, the continuous transmission of Quiet symbols is not used to transfer information. Instead, the continuous transmission of Quiet symbols is used as a signal to break a connection and restart the connection initialization sequence. The utilization of the line states for signalling is:

Quiet Line State	Used to break link and restart connection initialization sequence. Quiet Line State is also used to acknowledge the termination of a Trace process.
Halt Line State	Used to represent a bit of information.
Master Line State	Used to represent a bit of information. Master Line State is also used to propagate a Trace from the Active state.
Idle Line State	Used to separate information bits and provide clock synchronization.

These PHY line states are used to signal information and provide a handshaking sequence. All occurrences of the line states QLS, HLS, MLS, and ILS that cause PCM state transitions shall be recognized. This is required for correct operation of the PCM signalling.

9.4.6 Link Confidence Test (LCT)

The Link Confidence Test is used to test a link to determine if the link quality is adequate for ring operation, and shall detect major link quality problems. The Link Confidence Test is executed during the bit signalling for PCM connection establishment. It is a cooperative test between the two Ports, and the connection is not completed until the Link Confidence Test is passed.

The duration of the Link Confidence Test is determined by the values exchanged by bit 4 and bit 5 of the PCM signalling sequence. Each side of a connection signals a desired test duration, and the longer duration of the two is used for the Link Confidence Test. Four durations are defined:

Name	Duration	When Used
Short	LC_Short	No recent history of link errors and connection is not to be withheld
Medium	LC_Medium	Following a failure of LCT and the connection is not to be withheld
Long	LC_Long	Following rejection of a connection by LEM and the connection is not to be withheld
Extended	LC_Extended	To withhold a connection

Each Port sources data, which the neighbour node checks for errors. Examples are:

a) Transmitting Idle symbols and counting link errors on reception using the Link Error Event detector;

b) Transmitting PDUs and counting link errors on reception using the Link Error Event detector;

c) Transmitting PDUs and counting Frame Check Sequence (FCS) errors. Using this method, the criterion for passing the test will depend upon the number of frames received;

d) Looping back symbols received from the other end of the connection and counting link errors on reception using the Link Error Event detector.

The sixth bit transmitted in PCM signalling indicates the intent to use a MAC for the Link Confidence Test. When either side of the connection signals that a MAC Link Confidence Test is intended, then a ring shall be formed for the test. In this case, if a MAC is not inserted by a station, then the PHY shall be configured to repeat data through the repeat filter. When test data is transmitted by a MAC, it shall conform to the MAC protocol and be transmitted using a token as in normal ring operation. If neither side signals the intent to use a MAC, the data sourced shall be made up of valid PHY symbols that produce Idle Line State or Active Line State.

9.4.6.1 Link Confidence Test policies

The Link Confidence Test shall be repeated until it is passed. This is necessary to correctly proceed when there are multiple bypassed nodes joining in a random order or when a connector is slowly inserted. Failure of the Link Confidence Test shall cause PCM to return to the Break state, where the connection is reinitialized. Notification shall be given to SMT of the failure of the Link Confidence Test.

9.4.6.2 Link Confidence Test actions

The Link Confidence Test shall transmit Idle symbols or PDUs for the duration of the test. The reception of Master line state shall signal the successful completion of the Link Confidence Test, while the reception of Halt line state shall signal the failure of the Link Confidence Test. Reception of Quiet line state shall cause abortion of the Link Confidence Test.

The Link Confidence Test shall be performed by checking for excessive incoming errors or inappropriate line states for a period of time. Error events may be inadvertently reported upon the reception of Halt line state or Master line state at the termination of the Link Confidence Test. The Link Confidence Test shall be failed if any link error has occurred, excluding any inadvertent errors introduced as described above. The Link Confidence Test may use a more stringent threshold value to prevent marginal links from being added to the ring. The Link Confidence Test shall conclude by transmitting the test results to the opposing Port at the other end of the link via PCM signalling.

Successful completion of the Link Confidence Test shall be indicated by the clearing of the PC_LCT_Fail variable. Failure of the Link Confidence Test shall be indicated by the incrementing of the PC_LCT_Fail variable. The PC_LCT_Fail variable shall retain the number of consecutive failures of the Link Confidence Test. The value of the LEM_Fail flag is retained across successive failures of the Link Confidence Test.

9.4.7 Link Error Monitor (LEM)

9.4.7.1 Link Error Monitor objectives

The Link Error Monitor (LEM) examines the link error rate (LER) of an active link using PHY layer services and indicates to other SMT entities physical links having an inadequate bit error rate , e.g. due to a marginal link quality, link degradation, or connector unplugging. Other SMT entities may use these services for fault diagnostics or fault isolation purposes. These LEM fault detection services shall be provided for all links, even when a MAC entity is detecting errors on the same link as well.

The LEM function complements the initial Link Confidence Test, performed by PCM, and the PM_SIGNAL.indication service provided by PMD. The LEM function is required for every Port in a station or concentrator. The two LEM inputs are the cutoff and alarm thresholds. The required outputs of the LEM include an aggregate count and a Link Error Rate (LER) estimate.

The components of the LEM are: 1) a detector, 2) an accumulator, 3) a threshold test, and 4) an estimator. The specified inputs and outputs shall be provided to ensure interoperability and do not imply any particular implementation of the LEM components. For example, a long-term estimate may be derived from either the time between errors or a count of error occurrences with a given sample time. The algorithm used depends upon the particular implementation of the accumulator function at the node.

9.4.7.2 Link error event detector

The detector function identifies link error events. It is implementation dependent and has no standard specified input/output interfaces available to local or remote management. The detector uses the PHY SM_PH_STATUS.indication service interface to monitor the Link Error Rate of the associated link continuously while its PCM is in the Active state. The assertion of Line State Unknown (LSU) by a PHY is used as indication of a link error event, except for when the LSU is caused by a normally functioning preceding PHY, e.g. the receipt of a JK symbol pair, or the receipt of Halt symbols from a Repeat Filter.

NOTE 13 – Note 4 in the current part of ISO/IEC 9314 on PHY states the intention that a J symbol not followed by a K symbol be treated as a Violation and that future revisions of PHY will require such action. However, since this was in a note and not expressly stated in all of the appropriate places in the ISO/IEC 9314-1, it does not constitute a requirement and may have been ignored by an implementer. Should this be the case, such a station may output an IJI sequence which may then legitimately be recognized as a link error event as defined herein.

Link error events shall include:

a) Transitions from ILS to LSU with the duration of LSU exceeding 2 symbol times (80 ns);

b) Transitions from ALS to LSU with the duration of LSU exceeding 8 symbol times (320 ns).

An implementation may also optionally interpret (using optional PHY services) the following events as link error events:

a) Other transitions from ALS to LSU provided that these transitions are not caused by a Halt symbol, and the duration of LSU exceeds 2 symbol times;

b) Transitions from ALS to LSU of the duration not exceeding 2 symbol times, provided such transitions are not caused by a JK symbol pair, or by the actions of the Elasticity Buffer or the Decode function in the same PHY;

c) Transitions from ILS to LSU of the duration not exceeding 2 symbol times, provided such transitions are not caused by a JK symbol pair, or by the actions of the Elasticity Buffer or the Decode function in the same PHY.

9.4.7.3 Link error event accumulation

The accumulation function counts or time stamps link error events and produces the aggregate count LEM_Ct. It provides the data for the threshold test and the statistical estimator components. The output available to local and remote management is the LEM_Ct:

LEM_Ct The aggregate link error count, set to zero only on station initialization.

9.4.7.4 Link Error Rate threshold test

The threshold test compares the current Link Error Rate (LER) to preset cutoff and alarm Link Error Rate thresholds. The Link Error Rate is based on a statistical relationship between the number of errors observed in a given interval and the current Link Error Rate. This relationship can be expressed as:

$$\mathrm{LER} \geq \frac{N}{T \times 125 \times 10^{6}}$$

where

LER is the current LER rate:

N is the number of link error events observed in a T second time interval.

The inputs to the LER threshold test are:

LER_Cutoff The link error rate at which a connection will be flagged as faulty. LER_Cutoff may take on a value from 10^{-4} to 10^{-15}. This attribute is reported external to LEM as the absolute value of the base 10 logarithm of the LER_Cutoff value. The default value of LER_Cutoff is 10^{-7}.

NOTE 14 – If an LER_Cutoff smaller than the default value is used, then LC_Long at each end of the physical connection should be long enough to prevent re-establishment of the faulty connection.

LER_Alarm The link error rate at which a link connection exceeds a preset alarm threshold. LER_Alarm may take on a value from 10^{-4} to 10^{-15}. This attribute is reported external to LEM as the absolute value of the base 10 logarithm of the LER_Alarm value. The default value of LER_Alarm is 10^{-8}.

The outputs of the LER threshold test provided to local management are:

LEM_Fail A flag set when the LER_Estimate exceeds the Port's LER_Cutoff. The LEM_Fail flag is used to remove connections with excessive Link Error Rates and to establish the Link Confidence Test length.

LEM_Reject_Ct The number of times a link is removed due to exceeding the threshold test.

9.4.7.5 Link Error Monitor estimator

The LEM estimator provides a long-term average link error rate estimate, available to local and remote management. The output of the LEM estimator is:

LER_Estimate A long-term average link error rate. LER_Estimate ranges from 10^{-4} to 10^{-15}. This attribute is reported external to LEM (fddiPORTLerEstimate) as the absolute value of the base 10 logarithm of the LER_Estimate value.

9.4.8 Path Test

A Path Test function is required in FDDI to determine if a problem on an FDDI ring is in the node being managed. Fault recovery places a heavy reliance on the Path Test function to locate faulty MACs and data Paths. An implementation of a Path Test which is not complete compromises the fault recovery for the entire FDDI network.

The Path Test is implementation dependent and is not specified in this part of ISO/IEC 9314. It is recommended that the Path Test include the following:

a) testing of all accessible data Paths within the node;

b) loopback testing of the PHY entities as close as possible to the PMD interface;

c) confirmation of the operational parameters provided to the MAC;

d) testing of the MAC recovery process for this node including the resolution of the Beacon and Claim processes.

9.4.9 Trace function

The Trace function provides a recovery mechanism for stuck Beacon conditions on the FDDI ring. Whereas Physical Connection Management will provide recovery from most physical faults that occur between two nodes, the Trace function is intended to provide recovery from a stuck Beacon condition that cannot be localized to a single link. With such a static beaconing condition, the fault domain is located between the beaconing MAC and its nearest upstream neighbour MAC. The Trace function causes all stations and concentrators in the suspected fault domain to leave the ring and Path Test so that the fault may be localized.

9.4.9.1 Trace initiation

The Trace function is initiated by RMT as a result of a MAC that has been beaconing continuously for an extended period of time. This interval is sufficiently long to indicate that a static Beacon condition is present. Upon initiation, the MAC sets a Trace_Prop flag indicating that the Trace is to be propagated upstream through the Port that is feeding the MAC input. The Entity Coordination Management (ECM) possesses the configuration information and propagates the Trace to the appropriate Port by issuing a PC_Trace signal to the associated PCM. The reception of a Trace_Prop by the ECM causes it to enter the Trace state signifying that a Trace is in progress.

9.4.9.2 Trace propagation

When a PCM receives a PC_Trace signal, it transitions to the Trace state and transmits alternating Halt and Quiet symbols. The Port at the other end of the link will receive MLS and its associated PCM will set the Trace_Prop flag to indicate that the Trace is to be propagated upstream.

9.4.9.3 Trace termination

When a Trace_Prop flag is set and a MAC is directly upstream of the entity that set the Trace_Prop flag, the upstream MAC has been reached and the Trace is terminated. This causes the Path_Test variable to be set to Pending to indicate that a Path Test will be performed after a Leave sequence is completed. Termination of the Trace occurs when the ECM enters the Leave state.

To provide for recovery in the event of a failed Trace propagation, the ECM allows the Trace state to be maintained for no greater than the time required for the Trace to traverse the entire network.

The Trace is also terminated when a Disconnect is issued in a station or concentrator that is part of the Trace process. This is necessary so that the disconnect signal is recognized from all states.

9.4.9.4 Trace acknowledgement

When the ECM of a station or concentrator enters the Leave state, Quiet symbols are transmitted from all Ports to indicate that the node is leaving the network. The transmission of Quiet symbols indicates that the Trace has terminated and serves as an acknowledgement. The reception of QLS by a PCM in the Trace state signals that the Trace is complete and the node can perform a Path Test. As each node in the Trace domain successively receives QLS and then leaves for a Path Test, the Trace acknowledgement propagates back to the station or concentrator that originated the Trace.

9.5 Entity Coordination Management (ECM)

9.5.1 ECM functional description

Entity Coordination Management, shown in figure 10, controls the optional optical bypass switch of the Physical Media Dependent (PMD) layer and signals the Physical Connection Management (PCM) when the media is available. The Entity Coordination Management starts the PCMs for the A and B Ports in the node when the optical bypass switching is complete. Additionally, in concentrators, ECM starts the PCMs associated with the M Ports. Individual PCMs may be enabled and disabled while in the In state.

The ECM also coordinates the Trace function within the node. This includes propagating the Trace and terminating the Trace. The ECM initiates the Path Test after the completion of the Trace to localize the suspected fault. The Path Test is intended to test Paths within the node and MACs involved in the fault domain. This includes the MAC initiating the Trace and nearest upstream neighbour MAC.

9.5.2 Detailed ECM description

9.5.2.1 State EC0:OUT

When CMT is initialized, the ECM enters the Out state. In this state, the ECM waits for a Connect request from SMT. A successful completion of a Path Test is required to proceed out of the Out state. In the Out state, if an optical bypass switch is present it is in a bypassed state.

EC(01): Connect without Bypass - A transition directly to the In state occurs when Connect is issued, the last Path Test was passed, and no optical bypass switch is present.

EC(05): Connect with Bypass - A transition to the Insert state occurs when Connect is issued, the last Path Test was passed, and an optical bypass switch is present.

9.5.2.2 State EC1:IN

The In state is the normal state for a completed connection. Upon entry into the In state, PC_Start signals are sent to the PCM state machines.

EC(11a): Start Hold - When Hold is set and the NO_Flags are cleared the RE_Flag is cleared to start holding.

EC(11b): End Hold - When both NO_Flags are set the RE_Flag is set to end holding and allow normal recovery.

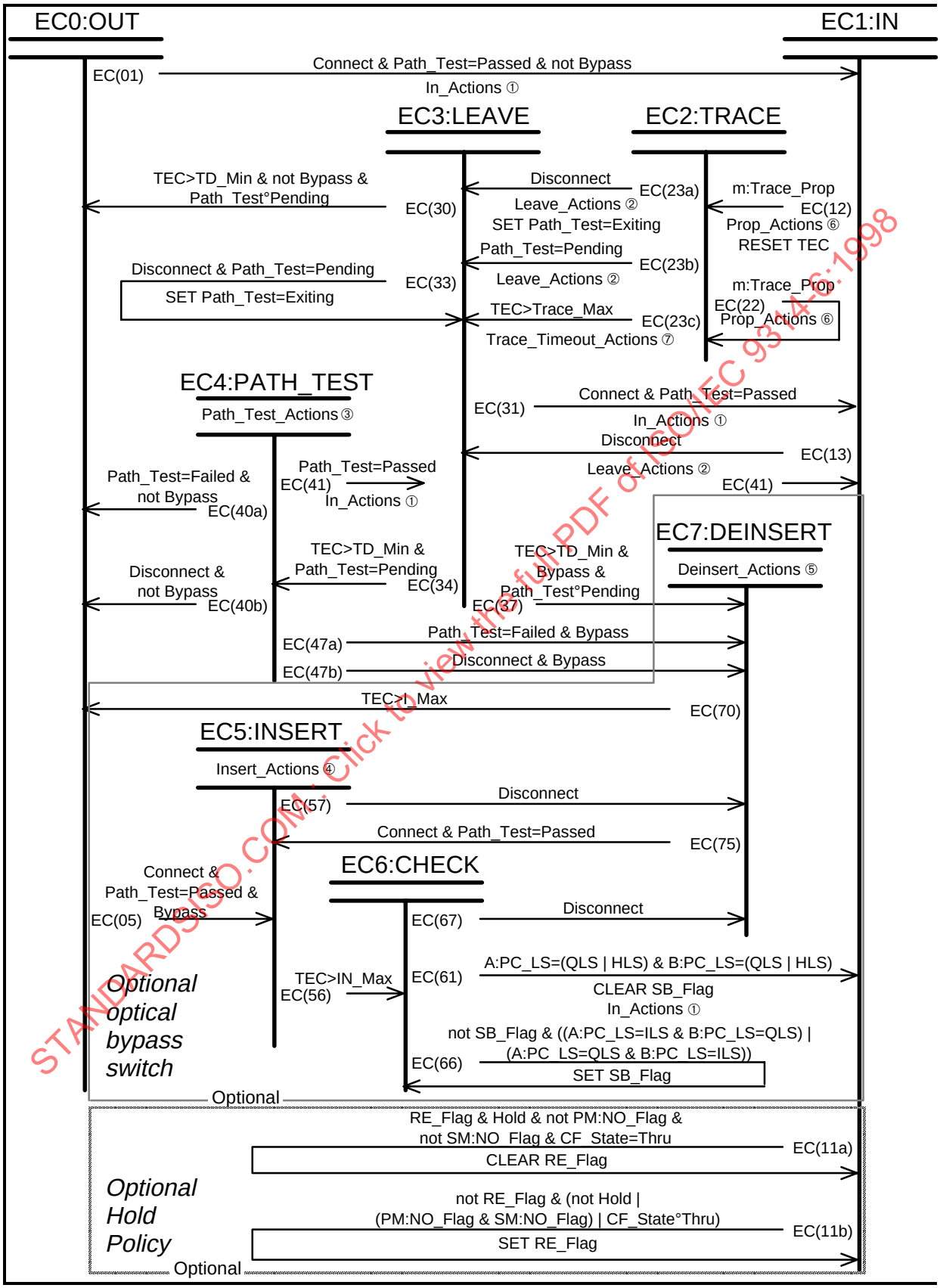


Figure 10 – Entity Coordination Management (ECM) state diagram (Part 1 of 2)

```
Entity Coordination Management footnotes:
1.  In_Actions:
          CLEAR *:Trace_Prop
          SET RE_Flag
          *:PC_Start
2.  Leave_Actions:
          *:PC_Stop
          RESET TEC
3.  Path_Test_Actions:
          SET Path_Test=Testing
          Perform Path Test
          IF Path Test Passed
               THEN SET Path_Test=Passed
                    CLEAR <all Paths>:TraceInitiated
                    CLEAR <all Paths>:TracePropagated
                    CLEAR <all Paths>:TraceTerminated
                    CLEAR <all Paths>:TraceTimeout
               ELSE SET Path_Test=Failed
4.  Insert_Actions:
          SM_PM_BYPASS.request(Insert)
          RESET TEC
5.  Deinsert_Actions:
          SM_PM_BYPASS.request(Deinsert)
          RESET TEC
6.  Prop_Actions:
          IF entity m is a MAC
               THEN SET <m:CurrentPath>:TraceInitiated
          IF entity m input is connected to a MAC output
               THEN
                    BEGIN
                         SET <CurrentPath of MAC connected to m
                              input>:TraceTerminated
                         SET Path_Test = Pending
                    END
               ELSE
                    BEGIN
                         SET <Path connected to m input>:TracePropagated
                         SIGNAL <Port connected to m input>:PC_Trace
                    END
          CLEAR m:Trace_Prop
7.  Trace_Timeout_Actions:
          *:PC_Stop
          RESET TEC
          SET Path_Test = Pending
          FOR <all Paths> DO
               IF Path:TraceInitiated | Path:TracePropagated
                    THEN SET Path:TraceTimeout
  NOTE
  A: and B: denote Port A and Port B, respectively.
  PM: denotes a MAC attached to the Primary Path.
  SM: denotes a MAC attached to the Secondary Path.
```

Figure 10 – Entity Coordination Management (ECM) state diagram (Part 2 of 2)

EC(12): Trace Propagation - A transition to the Trace state is made upon the reception of a Trace_Prop signal. The Prop_Actions are performed upon this transition. The Trace is propagated using the CF_State and fddiPATHConfiguration information.

EC(13): Disconnect - A transition to the Leave state is made if a Disconnect request is received while in the In state.

9.5.2.3 State EC2:TRACE

The Trace state is used to localize a stuck Beacon condition. TEC is reset upon initial entry into the Trace state.

EC(22): Trace Propagation - A transition back to the Trace state is made upon the recognition of a set Trace_Prop flag. The Prop_Actions are performed upon this transition. The Trace is propagated using the CF_State and fddiPATHConfiguration information.

EC(23a): Disconnect - A transition to the Leave state is made if a Disconnect request is received while in the Trace state. The Path_Test variable is set to Exiting so that a disconnection will be made and reconnections will not be allowed until a Path Test is performed.

EC(23b): Path Test requested - A transition to the Leave state is made if Path Test is set to Pending by a PCM while in the Trace state.

EC(23c): Time Out - A transition to the Leave state is made if the Trace state is retained for greater than Trace_Max time. The Path_Test variable is set to Pending on this transition and Trace_Timeout is signalled.

9.5.2.4 State EC3:LEAVE

The Leave state is present to allow sufficient time to break any existing connections. Upon entry into the Leave state, PC_Stop signals are sent to the PCM state machines and TEC is reset. The Leave state is held for a time of TD_Min.

EC(30): Leave without Bypass - A transition to the Out state is made when TEC>TD_Min, no bypass switch is present, and the Path_Test variable is not set to Pending. The delay of TD_Min in this transition ensures that the station or concentrator has completely deinserted before a transition is made to the Out state.

EC(34): Path Test - A transition to the Path_Test state is taken when TEC>TD_Min and the Path_Test variable is set to Pending.

EC(31): Connect - A transition to the In state is taken if a Connect request is received while the ECM is in the Leave state and the Path_Test variable is set to Passed.

EC(33): Disconnect - The Path_Test variable is set to Exiting when Disconnect is signalled while in the Leave state.

EC(37): Leave with Bypass - A transition to Deinsert is made after TEC>TD_Min when a bypass switch is present and the Path_Test variable is not set to Pending.

9.5.2.5 State EC4:PATH_TEST

The Path_Test state is entered upon the completion of the Trace function. In the Path_Test state, the station or concentrator performs a test of its entities and data Paths.

EC(40a): Failed without Bypass - A transition is made to the Out state when Path_Test is set to Fail and no optical bypass switch is present.

EC(40b): Disconnect without Bypass - A transition is made to the Out state when no optical bypass switch is present and a Disconnect is requested.

EC(41): Passed - A transition directly to the In state occurs when the Path_Test is set to Passed.

EC(47a): Failed with Bypass - A transition is made to the Deinsert state when the Path_Test is set to Fail and an optical bypass switch is present.

EC(47b): Disconnect with Bypass - A transition is made to the Deinsert state when an optical bypass switch is present and a Disconnect is requested.

9.5.2.6 State EC5:INSERT

The Insert state of the ECM allows for the switching time of the optical bypass switch. In the Insert state, an Insert request is sent to the optical bypass switch and TEC is reset. The ECM state remains as Insert until the optical bypass switch has completed switching.

EC(56): Switching Complete - A transition occurs to the Check state after IN_Max.

EC(57): Disconnect - A transition to the Deinsert state is made if a Disconnect request is received while in the Insert state.

9.5.2.7 State EC6:CHECK

The Check state of the ECM is used to confirm that both the primary and secondary optical bypass switches have switched.

EC(61): Switch Complete - A transition is made to the In state when QLS or HLS is observed on both A and B Ports. The SB_Flag is cleared on this transition.

EC(66): Switch Failed - This transition is made if line states are observed on both Ports that indicate that a stuck bypass switch may be present. The SB_Flag is set on this transition.

EC(67): Disconnect - A transition to the Deinsert state is made if a Disconnect request is received while in the Check state.

9.5.2.8 State EC7:DEINSERT

The Deinsert state allows time for the optical bypass switch to deinsert. An SM_PM_BYPASS.request(Deinsert) is issued upon entry into the Deinsert state.

EC(70): Deinsertion Complete - A transition to the Out state is made after TEC>I_Max.

EC(75): Connect - A transition is made to the Insert state when a Connect request is received and the Path Test was previously passed.

9.6 Physical Connection Management (PCM)

The Physical Connection Management (PCM) initializes the connection of neighbouring Ports and manages the signalling between Ports. PCM provides all the necessary signalling to

– initialize a connection
– withhold a marginal connection
– support maintenance.

The PCM is functionally divided into two entities: the PCM state machine and the PCM pseudo code. The PCM state machine contains all the state and timing information of the PCM and provides a signalling channel. The PCM pseudo code specifies the bits that are to be signalled by the PCM state machine and processes the bits received from the PCM at the other end of the link.

9.6.1 PCM functional description

The PCM state machine is shown in figure 11. The PCM state machine specifies the timing and state requirements for Physical Connection Management. The structure of the PCM state machine is independent of the bits of information provided by the PCM pseudo code. The state machine states and transitions are described in detail in this subclause.

9.6.1.1 Initialization sequence

Connection initialization starts with the reception of PC_Start from the Entity Coordination Management (ECM). PC_Start indicates that the physical media is available for communication. The PC_Start causes the PCM to enter the Break state where a continuous stream of Quiet symbols is transmitted to cause the Port at the other end of the link to stop signalling or transmitting data and enter the Break state. After sufficient Quiet symbols are transmitted, the reception of QLS or HLS causes the PCM to transition to the Connect state.

In the Connect state, a continuous stream of Halt symbols is transmitted for a sufficient time for clock acquisition by the receiving PHY. The reception of HLS causes the PCM to transition to the Next state. Upon this first transition to the Next state, the bit signalling counter is reset to n=0.

In the Next state, a continuous stream of Idle symbols is transmitted to separate the sending of bit synchronized signals. The Next state is terminated by the successful reception of ILS, the transmission of Idle symbols for a time sufficient for recognition, and the assertion of PC_Signal for the next bit to be transmitted.

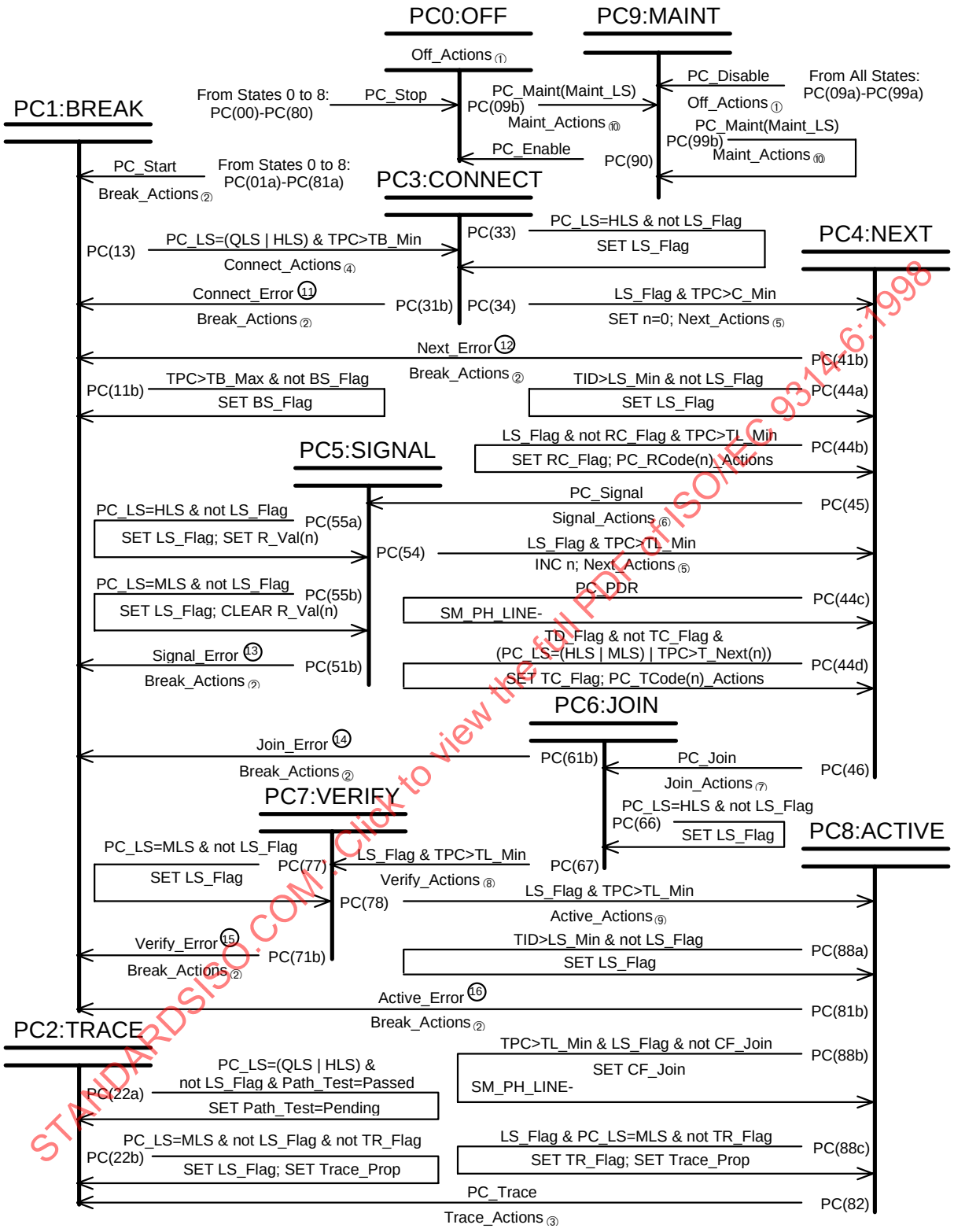


Figure 11 – Physical Connection Management (PCM) state diagram (Part 1 of 3)

```
1.  Off_Actions:
      SM_PM_CONTROL.request(Transmit_Disable)†
      SM_PH_Line-State.request(TRANSMIT_QUIET)
      CLEAR CF_Loop
      CLEAR CF_Join
      CLEAR BS_Flag
      SET PC_Withold = None
      SET Connect_State = Disabled
2.  Break_Actions:
      SM_PM_CONTROL.request(Transmit_Disable)†
      SM_PH_Line-State.request(TRANSMIT_QUIET)
      CLEAR CF_Loop
      CLEAR CF_Join
      CLEAR BS_Flag
      PC_Mode=N
      If PC_Withold = None
            THEN SET Connect_State = Connecting
3.  Trace_Actions:
      CLEAR LS_Flag
      SM_PH_Line-State.request(TRANSMIT_MASTER)
4.  Connect_Actions:
      CLEAR LS_Flag
      CLEAR BS_Flag
      SM_PM_CONTROL.request(Transmit_Enable)†
      SM_PH_Line-State.request(TRANSMIT_HALT)
5.  Next_Actions:
      CLEAR LS_Flag, RC_Flag, TC_Flag, TD_Flag
      SM_PH_Line-State.request(TRANSMIT_IDLE)
6.  Signal_Actions:
      CLEAR LS_Flag
      IF T_Val(n)
            THEN SM_PH_Line-State.request(TRANSMIT_HALT)
            ELSE SM_PH_Line-State.request(TRANSMIT_MASTER)
7.  Join_Actions:
      CLEAR LS_Flag
      SM_PH_Line-State.request(TRANSMIT_HALT)
8.  Verify_Actions:
      CLEAR LS_Flag
      SM_PH_Line-State.request(TRANSMIT_MASTER)
9.  Active_Actions:
      CLEAR LS_Flag, TR_Flag, LEM_Fail
      SM_PH_Line-State.request(TRANSMIT_IDLE)
      SET Connect_State = Active
10. Maint_Actions:
      IF Maint_LS=QUIET
            THEN SM_PM_CONTROL.request(Transmit_Disable)†
            ELSE SM_PM_CONTROL.request(Transmit_Enable)†
      SM_PH_Line-State.request(Maint_LS)
11.  Connect_Error:
      (PC_LS=ILS & not LS_Flag) or optionally
      PC_LS=MLS
12.  Next_Error:
      PC_LS=QLS or
      (n > 4 & T_Val(4) & T_Val(5) & (not WC_Flag) &
      Connection_Policy(PC_Type,PC_Neighbour)) or
       (not LS_Flag & n>0 &
        (TPC>T_Out or optionally
          (PC_LS=MLS & R_Val(n-1)) or
          (PC_LS=HLS & not R_Val(n-1)))) or optionally
      TNE>NS_Max
† This primitive is not required for all PMD implementations.
13.  Signal_Error:
      PC_LS=QLS or
      (not LS_Flag & TPC>T_Out) or  optionally
      TNE>NS_Max or optionally
      (LS_Flag &
       ((PC_LS=MLS & R_Val(n)) or (PC_LS=HLS & not R_Val(n))))
14.  Join_Error:
      PC_LS=QLS or
      (not LS_Flag & TPC>T_Out) or
      (LS_Flag & PC_LS=ILS) or optionally
      TNE>NS_Max
```

Figure 11 – Physical Connection Management (PCM) state diagram (Part 2 of 3)

```
15.  Verify_Error:
      PC_LS=QLS or
      (not LS_Flag & (PC_LS=ILS or TPC>T_Out)) or optionally
      (LS_Flag & PC_LS=HLS) or optionally
      TNE>NS_Max
16.  Active_Error:
      not TR_Flag &
      (PC_LS=HLS or WC_Flag or
       (RE_Flag &
        (PC_LS=QLS or LEM_Fail or TNE>NS_Max or
         (not LS_Flag & TPC>T_Out))))
17.  Precedence:
      a.    PC_Disable shall take precedence over PC_Stop.
      b.    PC_Stop shall take precedence over all conditions except PC_Disable.
      c.    Transitions to State PC1:BREAK shall take precedence over all conditions
      except PC_Stop and PC_Disable.
18.  PC_React:
      Transitions to State PC1:BREAK triggered by receipt of QLS shall occur within
      PC_React time after receipt of QLS.
19.  TPC:
      TPC shall be reset on every state transition.  This includes transitions where
      the target state is the same as the initial state.
```

Figure 11 – Physical Connection Management (PCM) state diagram (Part 3 of 3)

Signalling of information is done in the Signal state. The Signal state is maintained until HLS or MLS has been received and a sufficient length of Halt symbols or alternating Halt and Quiet symbols has been transmitted for recognition.

The PHY line states are used to signal information and provide handshaking during the signalling. All occurrences of the line states, QLS, HLS, MLS, and ILS, that cause PCM state transitions shall be recognized. This is required for correct operation of the PCM signalling.

The processing of the signalled bits is performed in the Next state. The PCM pseudo code describes the processing necessary for each bit. While in the Next state with bit counter of *n*, the PCM pseudo code processes the *n*-1 bit received from the other end of the link. After processing the last bit received, the pseudo code determines the next bit to be transmitted or if CF_Loop or PC_Join is to be asserted. If a bit is to be transmitted, the PCM pseudo code issues PC_Signal to cause a transition to the Signal state. The PCM state machine alternates between the Next and Signal states until all of the required connection initialization signals have been sent.

After all signalling has been completed successfully, the PC_Join signal is issued. Signalling of PC_Join causes the PCM to progress through the Join and Verify states to the Active state. Failure to receive the unique HLS-MLS-ILS sequence necessary for this progression shall cause the connection attempt to fail and return to the Break state. The signalling of PC_Join causes the CF_Join flag to be set in the CFM state machine so that a MAC is attached to the associated PHY.

9.6.1.2 Maintenance support

Support of maintenance functions is provided by the presence of a maintenance state where a specified line state shall be transmitted. The maintenance state allows CMT to force the PCM of the neighbouring Port into a known state manually. Line faults may be traced when both end nodes are in known states and do not change state due to line noise.

In addition, useful diagnostic properties are inherent in the PCM state machine. Consider the continuous transmission of QLS to a Port with its PCM in an unknown state:

a) A Port with its PCM in the Off state shall transmit continuous Quiet symbols in response;

b) The PCM in a Port attempting a connection shall reach the Connect state where the Port shall transmit Halt symbols continuously in response.

Thus, by the transmission of Quiet symbols to a Port with an unknown PCM state, the PCM status can be determined.

The steady-state properties for the state machine on a link with a broken fibre are also useful. The continuous reception of QLS that is received from a broken fibre causes a PCM to progress to the Connect state where it continuously transmits Halt symbols. The reception of this continuous stream of

Halt symbols causes a Port to reach a steady-state condition of continuously transmitting Idle symbols with its PCM in the Next state. Thus, a link break may be localized by observing whether QLS or ILS is seen on the link in addition to HLS in the other direction after the PCM has reached the above steady states.

9.6.2 Detailed PCM description

Required transition conditions for PCM are described in this clause. Optional additional transition conditions are specified in the PCM footnotes in figure 11, but are not described in this clause.

9.6.2.1 General transitions

The following transitions may be made from multiple states. They are made general so that they may be initiated externally without regard to the state of the PCM signalling.

PC(00)-PC(80): PC_Stop - Enter the Off state when PC_Stop is directed while in any state except the Maint state. A PC_Start shall be required to restart the PCM state machine.

PC(01a)-PC(81a): PC_Start - Enter the Break state when PC_Start is directed. This transition may be taken while in any state except the Maint state.

PC(09a)-PC(99a): PC_Disable - Enter the Maint state when PC_Disable is directed while in any state. A PC_Enable shall be required to remove the PCM from the Maint state.

9.6.2.2 State PC0:OFF

The Off state is the initial state of the PCM state machine. In the Off state, a continuous stream of Quiet symbols is transmitted. The Off state is entered when PC_Stop is detected while in any state in the PCM state machine except the Maint state. The PCM state machine remains in the Off state until a PC_Start or a PC_Maint is detected.

PC(09b): Maintenance - Enter the Maint state when PC_Maint*(Maint_LS) is directed from the Off state. Transmit the line state specified by Maint_LS.

9.6.2.3 State PC1:BREAK

The Break state is the entry point in the start of a PCM connection. In the Break state, a continuous stream of Quiet symbols is transmitted to force the other end of the connection to break any existing connection and restart the connection initialization sequence. The Break state is entered by the detection of a PC_Start, the reception of QLS, or the expiration of TPC>T_Out. The detection of PC_Start signifies that the connection shall be restarted, while the reception of QLS indicates either that there is a break in the connection or that the other end has started a reinitialization of the connection. The expiration of TPC>T_Out indicates that the signalling has halted and shall be restarted.

PC(11b): Stuck in Break - Set the BS_Flag when the PCM remains in the Break state for greater than TB_Max time.

PC(13): Connect - Enter the Connect state when QLS or HLS is being received after QLS has been transmitted for TB_Min. Clear LS_Flag on this transition.

9.6.2.4 State PC2:TRACE

The Trace state is used to localize a stuck Beacon condition. In the Trace state, a continuous stream of alternating Halt and Quiet symbol pairs is transmitted. The Trace state is entered by the detection of PC_Trace in the Active state. The Trace state is kept until the ECM starts a Path Test.

PC(22a): Trace Acknowledged - Set the ECM Path_Test variable to Pending if QLS or HLS is received while in the Trace state and the LS_Flag is not set.

PC(22b): Trace Received - Set the LS_Flag and Trace_Prop flags if MLS is received in the Trace state with the LS_Flag and TR_Flag flags not set.

9.6.2.5 State PC3:CONNECT

The Connect state is used to synchronize the ends of the connection for the signalling sequence. In the Connect state, a continuous stream of Halt symbols is transmitted. The Connect state is entered from the Break state when QLS or HLS is detected.

PC(31b): Restart - A transition to Break from Connect is made if ILS is received before HLS is received. The condition on the reception of ILS prevents deadlocks from occurring. This transition may optionally occur for other invalid input conditions.

PC(33): Line State Flagged - Set LS_Flag to indicate that HLS has been received while in the Connect state.

PC(34): Neighbour Connecting - The transition to Next occurs when the opposing PCM is in the Connect or Next states. HLS is transmitted for a length of time C_Min to assure that signal acquisition is achieved. Start the count of the signalled bits at n=0 and clear LS_Flag, RC_Flag, TC_Flag, and TD_Flag.

9.6.2.6 State PC4:NEXT

On initial entry into the Next state, a continuous stream of Idle symbols is transmitted. While in the Next state, either a continuous stream of Idle symbols or the symbol stream presented to the PH_DATA.request interface is transmitted as specified by the most recent argument to the SM_PH_LINE-STATE.request primitive. The Next state is used to separate the signalling performed in the Signal state and to transmit PDUs while MAC Local Loop is performed.

PC(41b): Restart - The reception of QLS indicates that the link is inactive. The Break state is entered to announce the reception of QLS or the termination of the conditions that were withholding the connection. This transition may optionally occur for other invalid input conditions.

PC(44a): Line State Flagged - Set the LS_Flag to indicate that ILS has been received while in the Next state.

PC(44b): Run RCode(n) - The receive pseudo code processing is started by this transition.

PC(44c): Transmit PDUs - PDUs may be transmitted from the Next state when sufficient ILS has been transmitted, ILS has been flagged, and PC_PDR has been issued.

PC(44d): Run TCode(n) - The transmit pseudo code processing is started by this transition only when its execution is delayed. The transmit pseudo code is started at the completion of the receive pseudo code when its execution is not delayed. This transition occurs when T_Next(n) time has passed since the last transition or when the line state changes from ILS to HLS or MLS. The TC_Flag ensures that this transition is not performed more than once for each entry into the Next state.

PC(45): Signal - Transition to the Signal state if PC_Signal has been issued. The issuing of PC_Signal indicates that the PCM pseudo code has been processed and the PCM has information ready to signal.

PC(46): Join - Transition to the Join state if PC_Join has been issued.

9.6.2.7 State PC5:SIGNAL

In the Signal state, individual bits of information are communicated across the connection by transmitting either a continuous stream of Halt symbols or a continuous stream of alternating Halt and Quiet symbol pairs. While in the Signal state, the PCM both transmits and receives bits of information at the same time. The Signal state shall only be entered from the Next state when a bit is ready to be transmitted.

PC(51b): Restart - The reception of QLS indicates a problem with the link or an inserting station or concentrator. Remaining in the Signal state longer than T_Out typically indicates that a line state detection was missed and the signalling sequence shall be restarted. This transition may optionally occur for other invalid input conditions.

PC(54): Next - Return to Next when PCM has received a response and has transmitted its signal for at least TL_Min. The bit counter *n* is incremented on this transition.

PC(55a): HLS Flagged - Flag the reception of HLS and indicate that the received bit was a one.

PC(55b): MLS Flagged - Flag the reception of MLS and indicate that the received bit was a zero.

9.6.2.8 State PC6:JOIN

The Join state is the first of three states in a unique sequence of transmitted symbol streams received as line states (HLS-MLS-ILS) that leads to an active connection. This sequence assures that both ends of a connection enter the Active state together at the completion of the signalling. In the Join state, a continuous stream of Halt symbols is transmitted.

PC(61b): Restart - The reception of QLS indicates a problem with the link or an inserting station or concentrator. Remaining in Join for longer than T_Out typically indicates that a line state detection was missed and the signalling sequence shall be restarted. This transition may optionally occur for other invalid input conditions.

PC(66): Line State Flagged - Set LS_Flag to indicate that HLS has been received while in the Join state.

PC(67): Verify - Transition to the Verify state when PCM has received a response and has transmitted its signal for at least TL_Min.

9.6.2.9 State PC7:VERIFY

The Verify state is the second state in the Path to the Active state. In the Verify state, a continuous stream of alternating Halt and Quiet symbol pairs is transmitted. The Verify state shall not be reached by a connection that is not synchronized.

PC(71b): Restart - The reception of QLS indicates a problem with the link or an inserting station or concentrator. Remaining in the Verify state longer than T_Out typically indicates that a line state detection was missed and the signalling sequence shall be restarted. This transition may optionally occur for other invalid input conditions.

PC(77): Line State Flagged - Set LS_Flag to indicate that MLS has been received while in the Verify state.

PC(78): Active - Transition to Active state when PCM has received a response and has transmitted its signal for at least TL_Min.

9.6.2.10 State PC8:ACTIVE

In the Active state, the Port is incorporated into the token ring Path. On initial entry into the Active, a continuous stream of Idle symbols is transmitted. In the Active state, either a continuous stream of Idle symbols or the symbol stream presented to the PH_DATA.request interface is transmitted as specified by the most recent argument to the SM_PH_LINE-STATE.request primitive.

PC(81b): Restart - Transition to Break if HLS or WC_Flag. If recovery is enabled, then transition to Break if QLS or LEM_Fail or extended noise is received or ILS has not been received for a period T_Out since entering the Active state. These conditions indicate a problem with the connection or a new station or concentrator joining the ring. This transition is not taken if the TR_Flag flag is set.

PC(82): Trace - Transition to the Trace state if PC_Trace is issued.

PC(88a): Line State Flagged - Set the LS_Flag to indicate that ILS has been received while in the Active state.

PC(88b): Ready to transmit - Signal CF_Join when ILS has been transmitted for a sufficient duration and ILS has been received. PDUs may now be transmitted on the ring.

NOTE 15 – An implementation should not enable the transmission of PDUs until Configuration Management (CFM) has properly reconfigured the CCEs due to the CF_Join being set.

PC(88c): Trace Received - Set the TR_Flag and Trace_Prop flag if MLS is received in the Active state with the LS_Flag set and the TR_Flag not set.

9.6.2.11 State PC9:MAINT

The Maint state is entered from the Off state upon the detection of PC_Maint(Maint_LS). In the Maint state, the symbol stream specified by the Maint_LS variable shall be forced. The PCM state machine is insensitive to the received line state while in the Maint state.

PC(90): Enable - A transition to the Off state is made when PC_Enable is signalled while in the Maint state.

PC(99b): Maintenance - Transmit the directed symbol stream Maint_LS in the Maint state.

9.6.3 PCM signalling

The PCM pseudo code provides the information to be communicated to the neighbouring PCM. The PCM pseudo code is divided into the transmit and receive Code subclauses. The interfaces to the PCM pseudo code are as shown in figure 11.

The transmit Code receives the information to be transmitted from other SMT entities and supplies the values for each bit to the PCM state machine and the Receive Code. The running of the transmit Code is initiated by the PC_TCode(n)_Actions signal. The transmit Code terminates with the issuing of PC_Signal when the information is ready to be signalled.

The Receive Code processes the transmitted and received information to support the prevailing connection policies. According to policy and the communicated information, the PCM Receive Code is capable of signalling PC_Start, PC_PDR, or PC_Join to the PCM state machine. The Receive Code provides the PC_Mode variable to the Configuration Management.

9.6.3.1 PCM signalled bits

T_Val(0) Escape Bit. T_Val(0) is not set for this sequence. The setting of T_Val(0) is reserved for future assignment by this part of ISO/IEC 9314. The following T_Val(n) assignments are used when T_Val(0) is not set or R_Val(0) is not set. Note that after signalling, R_Val(0) corresponds to T_Val(0) in the PCM at the other end of the connection.

T_Val(1,2) PC_Type. The value of PC_Type is encoded in T_Val(1) and T_Val(2) as follows:

PC_Type=A:	T_Val(1) not set	T_Val(2) not set
PC_Type=B:	T_Val(1) not set	T_Val(2) set
PC_Type=S:	T_Val(1) set	T_Val(2) not set
PC_Type=M:	T_Val(1) set	T_Val(2) set

The signalling of PC_Type is used to establish the mode of the connection and to prevent or detect topology problems.

T_Val(3) Port Compatibility. T_Val(3) is set if my topology rules include the connection of my Port to a Port of the type at the other end of the connection. The connection will be allowed if T_Val(3) is set or R_Val(3) is set. Connections that are not allowed are withheld later in the signalling sequence.

T_Val(4,5) LCT Duration. The requested Link Confidence Test duration is encoded into T_Val(4) and T_Val(5) as follows:

Name	**T_Val(4)**	**T_Val(5)**
Short	not set	not set
Medium	not set	set
Long	set	not set
Extended	set	set

T_Val(6) MAC Available for LCT. The setting of T_Val(6) indicates the intention to place a MAC at my end of the connection during the Link Confidence Test. The Link Confidence Test is performed following the signalling of T_Val(6).

T_Val(7) LCT Failed. T_Val(7) set indicates that the Link Confidence Test was failed by my end of the connection.

T_Val(8) MAC for Local Loop. The setting of T_Val(8) indicates that my end of the connection will provide a MAC for the MAC Local Loop. If T_Val(8) is not set and R_Val(8) is not set, then the MAC Local Loop is not performed. If T_Val(8) is set and R_Val(8) is not set, then my end of the connection has the option of not performing MAC Local Loop.

NOTE 16 – Whereas the Link Confidence Test is used to test the Link Error Rate over a timed interval, the MAC Local Loop may optionally be used to verify MAC recovery processes, token passing, and frame transmission and reception.

The MAC Local Loop is performed following the signalling of T_Val(8).

T_Val(9) MAC on Port Output. The setting of T_Val(9) indicates that my end of the connection intends to place a MAC in the output token Path of this Port. If this Port has PC_Mode set to Tree, the setting of this bit indicates that a MAC output will be connected to the PHY associated with this Port when the PCM becomes Active. If this Port has PC_Mode set to Peer, this bit indicates that a MAC output will be connected to the PHY associated with this Port when the node reaches a Thru state. This information may be used for physical ring map construction.

9.6.3.2 PCM pseudo code

```
1. PC_RCode(0)_Actions:
     PC_TCode(0)_Actions
2. PC_TCode(0)_Actions:
     CLEAR T_Val(0)
     SIGNAL PC_Signal
3. PC_RCode(1)_Actions:
     PC_TCode(1)_Actions
4. PC_TCode(1)_Actions:
     IF PC_Type=S or PC_Type=M
          THEN SET T_Val(1)
          ELSE CLEAR T_Val(1)
     SIGNAL PC_Signal
5. PC_RCode(2)_Actions:
     PC_TCode(2)_Actions
6. PC_TCode(2)_Actions:
     IF PC_Type=B or PC_Type=M
          THEN SET T_Val(2)
          ELSE CLEAR T_Val(2)
     SIGNAL PC_Signal
7. PC_RCode(3)_Actions:
     IF R_Val(1)
          THEN IF R_Val(2)
                    THEN SET PC_Neighbour=M
                    ELSE SET PC_Neighbour=S
          ELSE IF R_Val(2)
                    THEN SET PC_Neighbour=B
                    ELSE SET PC_Neighbour=A
     PC_TCode(3)_Actions
8. PC_TCode(3)_Actions:
     IF PC_Type=M & PC_Neighbour=M
          THEN CLEAR T_Val(3)
          ELSE IF Connection_Policy (PC_Type, PC_Neighbour)
                    THEN SET T_Val(3)
                    ELSE CLEAR T_Val(3)
     SIGNAL PC_Signal
9. PC_RCode(4)_Actions:
     IF PC_Type=M & PC_Neighbour=M
          THEN SET PC_Withhold=Port M to Port M
          ELSE  IF T_Val(3) or R_Val(3)
                    THEN
                         BEGIN
                              IF PC_Type=M or PC_Neighbour=M
                                   THEN  SET PC_Mode=T
                                   ELSE  SET PC_Mode=P
                              Reevaluate the selection criteria and the setting of the
                              WC_Flag (see 9.7.2.4.4)
                              IF WC_Flag
                                   THEN SET PC_Withhold=Requested Paths Not Available
                                   ELSE SET PC_Withhold=None
                         END
                    ELSE  SET PC_Withhold=Other Incompatible Port Types
     PC_TCode(4)_Actions
```