Information Processing Systems - Telecommunications and Information Exchange between Systems - Protocol for Exchange of Inter-domain Routeing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs

April 27, 1993

Contents

1. Scope and Field of Application 1

2. Normative References 1

3. Definitions 2
3.1 Reference Model Definitions 2
3.2 Network Layer Architecture Definitions . . . 2
3.3 Network Layer Addressing Definitions 2
3.4 Routeing Framework Definitions 3
3.5 Intra-domain Routeing Definitions 3
3.6 Additional Definitions 3

4. Symbols and Abbreviations 4
4.1 Data Unit Abbreviations 4
4.2 Addressing Abbreviations 4
4.3 Other Abbreviations 4

5. General Protocol Information 5
5.1 Inter-RD Topology 5
5.2 Routeing Policy 5
5.3 Types of Systems 7
5.4 Types of Routeing Domains 7
5.5 Routeing Domain Confederations 7
5.6 Routes: Advertisement and Storage 8
5.7 Distinguishing Path Attributes and RIB-Atts . 8
5.8 Selecting the Information Bases 8
5.9 Routeing Information Exchange 8
5.9.1 Internal Neighbor BIS 9
5.9.2 External Neighbor BIS 9
5.10 Routeing Domain Identifiers 10
5.11 Formats of RDIs, NETs, and NSAP Addresses 10
5.12 Design Objectives 10
5.12.1 Within the Scope of the Protocol 10
5.12.2 Outside the Scope of the Protocol . . . 11

6. Structure of BISPDUs 11
6.1 Header of BISPDU 11
6.2 OPEN PDU 12
6.3 UPDATE PDU 14
6.3.1 Path Attribute Encoding 15
6.3.2 Network Layer Reachability Information 20
6.4 IDRP ERROR PDU 20
6.5 KEEPALIVE PDU 21
6.6 CEASE PDU 22
6.7 RIB REFRESH PDU 22

7. Elements of Procedure 22
7.1 Naming and Addressing Conventions 22
7.1.1 Interpretation of Address Information . 22
7.1.2 NSAP Address Prefixes 22
7.2 Deployment Guidelines 23
7.2.1 Minimum Configuration of an RD 23
7.2.2 Deployment of ISs and ESs 23
7.3 Domain Configuration Information 23
7.4 Advertising NLRI Information 24
7.5 Receive Process 24
7.6 BIS-BIS Connection Management 25
7.6.1 BIS Finite State Machines 25
7.6.2 Closing a Connection 30
7.7 Validation of BISPDUs 30
7.7.1 Authentication Type 1 30
7.7.2 Authentication Type 2 30
7.7.3 Authentication Type 3 31
7.7.4 Sequence Numbers 32
7.7.5 Flow Control 32
7.8 Version Negotiation 34
7.9 Checksum Algorithm 34
7.10 Routeing Information Bases 34
7.10.1 Identifying an Information Base 34
7.10.2 Validation of RIBs 35
7.10.3 Use of the RIB REFRESH PDU 36
7.11 Path Attributes 36
7.11.1 Categories of Path Attributes 37
7.11.2 Handling of Distinguishing Attributes . 38
7.11.3 Equivalent Distinguishing Attributes . . 38
7.12 Path Attribute Usage 38
7.12.1 ROUTE_SEPARATOR 38
7.12.2 EXT_INFO 39
7.12.3 RD_PATH 39
7.12.4 NEXT_HOP 41
7.12.5 DIST_LIST_INCL 43
7.12.6 DIST_LIST_EXCL 43
7.12.7 MULTI-EXIT_DISC 44
7.12.8 TRANSIT DELAY 44
7.12.9 RESIDUAL ERROR 44
7.12.10 EXPENSE 45
7.12.11 LOCALLY DEFINED QOS 45
7.12.12 HIERARCHICAL RECORDING 45
7.12.13 RD_HOP_COUNT 46
7.12.14 SECURITY 46
7.12.15 CAPACITY 46
7.12.16 PRIORITY 47
7.13 Routeing Domain Confederations 47
7.13.1 RDC Policies 47
7.13.2 RDC Configuration Information 47
7.13.3 Detecting Confederation Boundaries . 47
7.14 Update-Receive Process 47
7.15 Information Consistency 48
7.15.1 Detecting Inconsistencies 48
7.16 Decision Process 48
7.16.1 Phase 1: Calculation of Degree of Preference 49
7.16.2 Phase 2: Route Selection 49
7.16.3 Phase 3: Route Dissemination 50
7.16.4 Interaction with Update Process 51
7.17 Update-Send Process 52
7.17.1 Internal Updates 52
7.17.2 External Updates 53
7.17.3 Controlling Routeing Traffic Overhead 53
7.18 Efficient Organization of Routeing Information 54
7.18.1 Information Reduction 54
7.18.2 Aggregating Routeing Information . . . 54

7.19 Maintenance of the Forwarding Information Bases . . . 57
7.20 Error Handling for BISPDUs . . . 57
7.20.1 BISPDU Header Error Handling . . . 57
7.20.2 OPEN PDU Error Handling . . . 58
7.20.3 UPDATE PDU Error Handling . . . 58
7.20.4 IDRP ERROR PDU Error Handling . . . 60
7.20.5 Hold Timer Expired Error Handling . . . 60
7.20.6 KEEPALIVE PDU Error Handling . . . 60
7.20.7 CEASE PDU Error Handling . . . 60
7.20.8 RIB REFRESH PDU Error Handling . . . 60

8. Forwarding Process for CLNS . . . 60
8.1 Forwarding to Internal Destinations . . . 61
8.2 Determining the NPDU-derived Distinguishing Attributes . . . 61
8.3 Matching RIB-Att to NPDU-derived Distinguishing Attributes . . . 61
8.4 Forwarding to External Destinations . . . 62

9. Interface to ISO 8473 . . . 63
9.1 Use of NLSP over ISO 8473. . . . 63

10. Constants . . . 64

11. System Management and GDMO Definitions 65
11.1 Name Binding . . . 65
11.2 Local BIS Managed Objects for IDRP . . . 65
11.3 Adjacent BIS Peer Managed Objects . . . 68
11.4 Attribute Definitions . . . 69
11.5 Parameter Definitions . . . 75
11.6 ASN.1 Modules . . . 76

12. Conformance . . . 81
12.1 Static Conformance for All BISs . . . 81
12.2 Conformance to Optional Functions . . . 81
12.2.1 Generation of Information in Reduced Form . . . 81
12.2.2 Generation of Well-known Discretionary Attributes . . . 81
12.2.3 Propagation of Well-known Discretionary Attributes . . . 82
12.2.4 Peer Entity Authentication . . . 82

Annex A. PICS Proforma . . . 83
A.1 Introduction . . . 83
A.2 Abbreviations and Special Symbols . . . 83
A.2.1 Status Symbols . . . 83
A.3 Instructions for Completing the PICS Proforma 83
A.3.1 General Structure of the PICS Proforma 83
A.3.2 Additional Information . . . 84
A.3.3 Exception Information . . . 84
A.3.4 Conditional Status . . . 84
A.4 Identification . . . 85
A.4.1 Implementation Identification: IDRP . . . 85
A.4.2 Protocol Summary: IDRP . . . 86
A.4.3 PICS Proforma: IDRP General . . . 86
A.4.4 PICS Proforma: IDRP Update Send Process . . . 86
A.4.5 PICS Proforma: IDRP Update Receive Process . . . 88
A.4.6 PICS Proforma: IDRP Decision Process . 88
A.4.7 PICS Proforma: IDRP Receive Process . 88
A.4.8 PICS Proforma: IDRP CLNS Forwarding 89
A.4.9 PICS Proforma: IDRP Authentication . . 89
A.4.10 PICS Proforma: IDRP Optional Transitive Attributes . . . 89
A.4.11 PICS Proforma: IDRP Well Known Discretionary Attributes . . . 90

Annex B. Bibliography . . . 92

Annex C. IDRP Checksum Generation Algorithm 92
C.1 Mathematical Notation . . . 92
C.2 Algorithm Description . . . 93

Annex D. Example of Authentication Type 2 . . . 94
D.1 Authentication Mechanism . . . 94

Annex E. Jitter Algorithm . . . 95

Annex F. Computing a Checksum for an Adj-RIB 96

Annex G. RIB Overload . . . 96

Annex H. Processor Overload . . . 97

Annex I. Formation of RDCs . . . 97
I.1 Forming a New Lower Level Confederation . 98
I.2 Forming a Higher Level Confederation . . . 98
I.3 Deleting a Lowest Level Confederation . . . 99
I.4 Deleting a Higher Level Confederation . . . 99

Annex J. Example Usage of MULTI-EXIT_DISC Attribute . . . 99

Annex K. Syntax and Semantics for Policy . . 102
1 K.1 Overview . . . 102
1 K.1.1 Preference Statement . . . 102
1 K.1.2 Aggregation Statement . . . 103
1 K.1.3 Distribution Statement . . . 104
1 K.2 Policy Configuration Language BNF . . . 105
1 K.2.1 PREF Statement BNF . . . 105
1 K.2.2 AGGR Statement BNF . . . 106
1 K.2.3 DIST Statement BNF . . . 106
1 K.2.4 Common BNF Symbols . . . 106
1 K.3 Simple Example . . . 109
1 K.3.1 Transit Domain 3 . . . 109
1 K.3.2 Policy Configuration Example . . . 110
1 K.3.3 Discussion . . . 111

Index . . . 115

Figures

1. Field of Application 2
2. Intermediate Routeing Domains and End Routeing Domains 4
3. Position of IDRP within Network Layer . . . 6
4. Inter-domain Routeing Components 7
5. Structure of the UPDATE PDU 15
1 6. Illustration of Authentication Types 1 and 3 31
7. Routeing Information Base 35
8. A Transitive Fully Connected Subnetwork . 42
9. Illustrative PICS for Conditional Items . . . 85
10. An Example of the Authentication Type 2 . 95
11. Jitter Algorithm for Timers 96
12. Example 1 Configuration 100
13. Example 2 Configuration 100
1 14. A Portion of an Internet 110

Tables

1. The IDRP Information Bases 9
2. BIS Finite State Machine 27
1 3. Path Attribute Characteristics 37
4 4. NPDU-Derived Attribute Set 62
5. IDRP-CL Primitives 64
6. Architectural Constants of IDRP 64
7. PICS Proforma for IDRP: Implementation Identification 85
8. PICS Proforma for IDRP: Protocol Summary 86
9. PICS Proforma for IDRP: General 86
10. PICS Proforma for IDRP: Update-Send . . . 87
11. PICS Proforma for IDRP: Update-Receive . 88
12. PICS Proforma for IDRP: Decision 88
13. PICS Proforma for IDRP: Receive 88
14. PICS Proforma for IDRP: CLNS Forwarding 89
15. PICS Proforma for IDRP: Peer Entity AUthentication 89
16. PICS Proforma for IDRP: Optional Transitive Attributes 89
17. PICS Proforma for IDRP: Generating Well Known Discretionary Attributes 90
18. PICS Proforma for IDRP: Propagating Well Known Discretionary Attributes 91
19. PICS Proforma for IDRP: Receiving Well Known Discretionary Attributes 92

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 the 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 10747 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information Technology*.

Introduction

This Protocol is one of a set of International Standards which facilitate the interconnection of open systems. They cover the services and protocols required to achieve such interconnection.

This Protocol is positioned with respect to other related standards by the layered structure defined in ISO 7498, and by the Network layer organization defined in ISO 8648. It is located at the top of the Network layer and relies on the services of ISO 8473. This protocol permits a routeing domain to exchange information with other routeing domains to facilitate the operation of the routeing and relaying functions of the Network Layer. It applies to the following categories of routeing, which are described in ISO TR 9575, making no distinction between them:

— Intra-Administrative Domain routeing between routeing domains
— Inter-Administrative Domain routeing between routeing domains.

Within the hierarchical relations between routeing protocols, as described in ISO TR 9575, this protocol is situated above the intra-domain routeing protocols. That is, this Inter-domain IS-IS protocol:

— maintains information about the interconnections between routeing domains, but does not require detailed information about their internal structures

— calculates path segments on a hop-by-hop basis

This protocol calculates path segments which consist of *Boundary Intermediate systems* and the links that interconnect them. An NPDU destined for an End system in another routeing domain will be routed via Intra-domain routeing to a Boundary Intermediate system (BIS) in the source routeing domain. Then, the BIS, using the methods of this inter-domain routeing protocol, will calculate a path to a Boundary Intermediate system in an adjacent routeing domain lying on a path to the destination. After arriving at the next routeing domain, the NPDU may also travel within that domain on its way towards a BIS located in the next domain along its path. This process will continue on a hop-by-hop basis until the NPDU arrives at a BIS in the routeing domain which contains the destination End system. The Boundary IS in this routeing domain will hand the incoming NPDU over to the domain's intra-domain routeing protocol, which will construct a path to the destination End system.

This inter-domain IS-IS routeing protocol places requirements on the type of information that a routeing domain must provide and on the methods by which this information will be distributed to other routeing domains. These requirements are intended to be minimal, addressing only the interactions between Boundary ISs; all other internal operations of each routeing domain are outside the scope of this protocol.[1]

The methods of this protocol differ from those generally adopted for an intra-domain routeing protocol because they emphasize the interdependencies between efficient route calculation and the preservation of legal, contractual, and administrative concerns. This protocol calculates routes which will be efficient, loop-free, and in compliance with the domain's local routeing policies. IDRP may be used when routeing domains do not fully trust each other; it imposes no upper limit on the number of routeing domains that can participate in this protocol; and it provides isolation between its operations and the internal operations of each routeing domain.

[1] For example, this Inter-domain routeing protocol does not mandate that a routeing domain run a particular intra-domain routeing protocol: it is local choice as to whether a domain implements a standard intra-domain protocol (such as ISO/IEC 10589) or a private protocol.

Information Processing Systems - Telecommunications and Information Exchange between Systems - Protocol for Exchange of Inter-domain Routeing Information among Intermediate Systems to Support Forwarding of ISO 8473 PDUs

1. Scope and Field of Application

This international standard specifies a protocol to be used by Boundary Intermediate systems (defined in 3.6) to acquire and maintain information for the purpose of routeing NPDUs between different routeing domains. Figure 1 illustrates the field of application of this international standard.

This international standard specifies:

— the procedures for the exchange of inter-domain reachability and path information between BISs
— the procedures for maintaining inter-domain routeing information bases within a BIS
— the encoding of protocol data units used to distribute inter-domain routeing information between BISs
— the functional requirements for implementations that claim conformance to this standard

The procedures are defined in terms of:

— interactions between Boundary Intermediate systems through the exchange of protocol data units
— interactions between this protocol and the underlying Network Service through the exchange of service primitives
— constraints on policy feasibility and enforcement which must be observed by each Boundary Intermediate system in a routeing domain

The boundaries of Administrative Domains are realized as artifacts of the placement of policy constraints and the aggregation of network layer reachability information; they are not manifested explicitly in the protocol. The protocol described in this international standard operates at the level of individual routeing domains. The establishment of administrative domains is outside the scope of this standard.

2. Normative References

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

ISO 7498: 1984, *Information Processing Systems - Open Systems Interconnection - Basic Reference Model*

ISO 7498/Add. 1: 1984, *Information Processing Systems - Open Systems Interconnection - Addendum to ISO 7498 Covering Connectionless-mode Transmission*

ISO 7498/Add. 3: 1984, *Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 3: Naming and Addressing*

ISO/DIS 7498/Add. 4[2]**,** *Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 4: OSI Management Framework*

ISO 8348: 1987, *Information Processing Systems - Telecommunications and Information Exchange between Systems - Network Service Definition*

ISO 8348/Add. 1: 1987, *Information Processing Systems - Telecommunications and Information Exchange between Systems - Addendum to the Network Service Definition Covering Connectionless-mode Transmission*

ISO 8348/Add.2: 1988, *Information Processing Systems - Telecommunications and Information Exchange between Systems - Addendum to the Network Service Definition Covering Network Layer Addressing*

[2] To be published

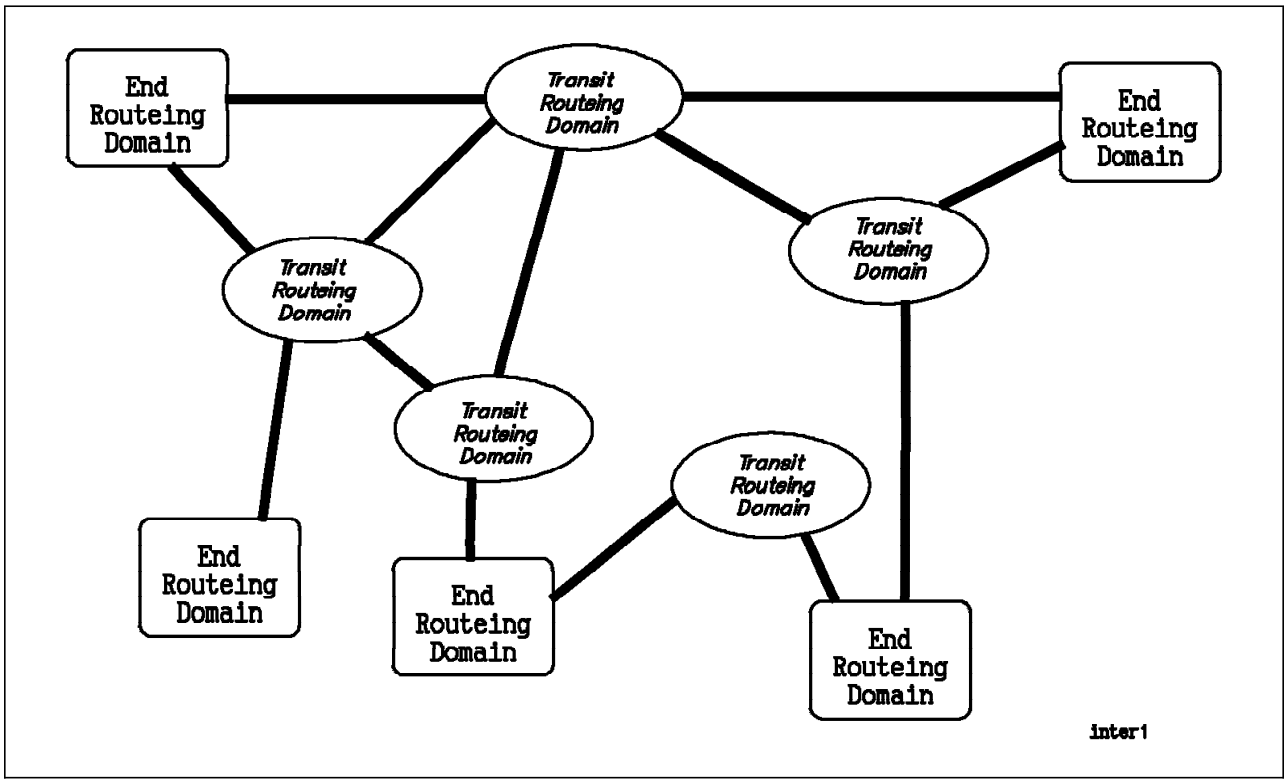

Figure 1. Field of Application. The Inter-domain Routeing Protocol operates between routeing domains; intra-domain routeing is not within its scope.

ISO 8473: 1988, *Information Processing Systems - Telecommunications and Information Exchange between Systems - Protocol for Providing the Connectionless-mode Network Service*

ISO 8648: 1988, *Information Processing Systems - Telecommunications and Information Exchange between Systems - Internal Organization of the Network Layer*

ISO TR 9577: 1991, *Information Processing Systems - Telecommunications and Information Exchange between Systems - Protocol identification in the Network Layer*

3. Definitions

For the purposes of this international standard, the following definitions apply.

3.1 Reference Model Definitions

This International Standard uses the following terms defined in ISO 7498:

a) (N)-entity title
b) Network entity
c) Network Layer
d) Network Protocol
e) Network Protocol Data Unit
f) Network relay
g) Network Service Access Point
h) Network Service Access Point Address
i) Real system
j) Routeing

3.2 Network Layer Architecture Definitions

This International Standard uses the following terms defined in ISO 8648:

a) End system
b) Intermediate System
c) Subnetwork

3.3 Network Layer Addressing Definitions

This International Standard uses the following terms defined in ISO 8348/AD2:

a) Subnetwork point of attachment

3.4 Routeing Framework Definitions

This International Standard uses the following terms defined in ISO 9575:

a) Administrative Domain
b) Common Domain
c) Fire wall
d) Routeing Domain

3.5 Intra-domain Routeing Definitions

This International Standard uses the following terms defined in IS 10589:

a) Adjacency
b) Link

3.6 Additional Definitions

For purposes of this International Standard, the following definitions are used:

Intra-domain IS-IS routeing protocol: A routeing protocol that is run between Intermediate systems in a single routeing domain to determine routes that pass through only systems and links wholly contained within the domain.

NOTE 1: Unless reference is made to a specific protocol, this term is used as a general designator, encompassing both private and internationally standardized protocols.

Inter-domain link: A real (physical) or virtual (logical) link between two or more Boundary Intermediate systems (see Figure 2). A link between two BISs in the same routeing domain carry both intra-domain traffic and inter-domain traffic; a link between two BISs located in adjacent routeing domains can carry inter-domain traffic, but not intra-domain traffic.

Boundary Intermediate system: An intermediate system that runs the protocol specified in this standard, has at least one inter-domain link attached to it, and may optionally have intra-domain links attached to it.

End Routeing Domain: A routeing domain whose local policies permit its BISs to calculate inter-domain path segments only for PDUs whose source is located within that routeing domain. There are two varieties of End routeing domains: stub and multi-homed. A stub ERD has inter-domain links to only one adjacent routeing domain, while a multi-homed ERD has inter-domain links to several adjacent routeing domains.

For example, the domains labelled as multi-homed ERDs in Figure 2 have policies which prohibit them from providing relaying functions; it is these policies, not the topology of their interconnections, that make them ERDs.

Transit Routeing Domain: A routeing domain whose policies permit its BISs to calculate inter-domain path segments for PDUs whose source is located either in the local routeing domain or in a different routeing domain. That is, it can provide a relaying service for such PDUs. See Figure 2 for an illustration of TRDs.

Adjacent RDs: Two RDs (″A″ and ″B″) are adjacent to one another if there is a at least one pair of BISs, one located in ″A″ and the other in ″B″, that are attached to each other by means of a real subnetwork.

RD Path: A list of the RDIs of the routeing domains and routeing domain confederations through which a given UPDATE PDU has travelled.

Routeing Domain Confederation: A set of routeing domains which have agreed to join together and to conform to the rules of clause 7.13 of this international standard. To the outside world, a confederation is indistinguishable from a routeing domain.

Nested RDCs: A routeing domain confederation ″A″ (RDC-A) is nested within RDC-B when all of the following conditions are satisfied simultaneously:

a) all members of RDC-A are also members of RDC-B
b) there are some members of RDC-B that are not members of RDC-A

Overlapping RDCs: A routeing domain confederation (RDC-A) overlaps RDC-B when all the following conditions are satisfied simultaneously:

a) there are some members of RDC-A that are also members of RDC-B, and
b) there are some members of RDC-A that are not members of RDC-B, and
c) there are some members of RDC-B that are not members of RDC-A.

Disjoint RDCs: Two routeing domain confederations, RDC-A and RDC-B, are disjoint from one another when there are no routeing domains which are simultaneously members of both RDC-A and RDC-B.

Policy Information Base: The collection of routeing policies that a BIS will apply to the routeing information that it learns using this International standard. It is not required that all routeing domains use the same syntax and semantics to express policy; that is, the format of the Policy Information Base is left as a local option.

Route Origin: Each route or component of an aggregated route has a single unique origin. This is the RD or RDC in which the route′s destinations are located.

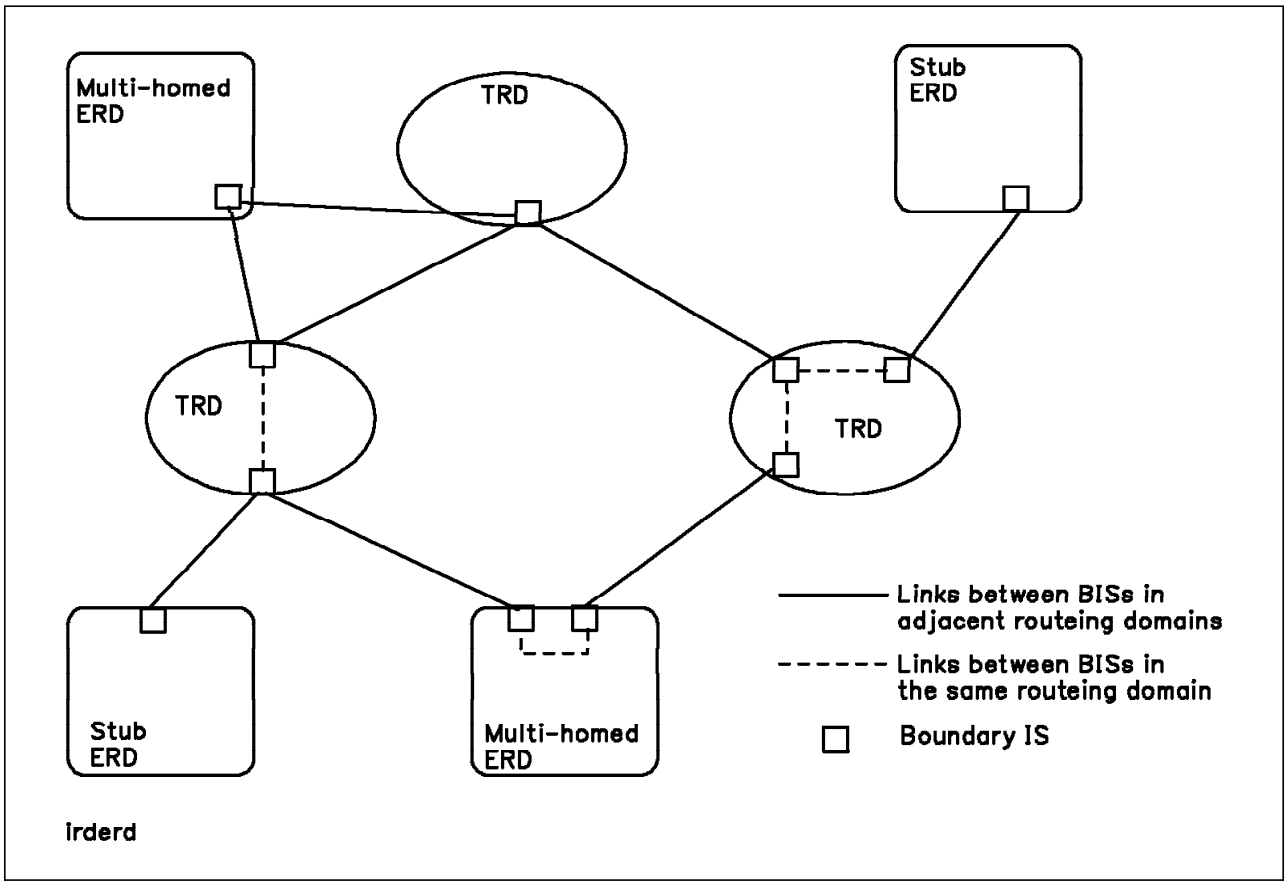

Figure 2. Intermediate Routeing Domains and End Routeing Domains. The classification of a routeing domain as an TRD or an ERD depends upon its relaying policies.

4. Symbols and Abbreviations

The symbols, acronyms, and abbreviations listed in the following clauses are used in this International Standard.

4.1 Data Unit Abbreviations

BISPDU Boundary Intermediate System PDU

DT PDU ISO 8473 Data Protocol Data Unit

ER PDU ISO 8473 Error Protocol Data Unit

NPDU Network Protocol Data Unit

NSDU Network Service Data Unit

PDU Protocol Data Unit

4.2 Addressing Abbreviations

AFI Authority and Format Identifier

DSP Domain Specific Part

IDI Initial Domain Identifier

IDP Initial Domain Part

LSAP Link Service Access Point

NET Network Entity Title

NPAI Network Protocol Address Information

NSAP Network Service Access Point

SNPA Subnetwork Point of Attachment

4.3 Other Abbreviations

BIS Boundary Intermediate System

CL Connectionless Mode

CLNS Connectionless Mode Network Service

CM Confederation Member

ERD End Routeing Domain

ES End System

FIB Forwarding Information Base

FSM Finite State Machine

IDRP Inter-domain Routeing Protocol (an acronym for the protocol described in this international standard)

IPI	Initial Protocol Identifier
MIB	Management Information Base
NLRI	Network layer reachability information
NLSP	Network layer security protocol
OSIE	OSI Environment
PCI	Protocol Control Information
PIB	Policy Information Base
QOS	Quality of Service
RDC	Routeing Domain Confederation
RDI	Routeing Domain Identifier
RIB	Routeing Information Base
SPI	Subsequent Protocol Identifier
SNICP	Subnetwork independent convergence protocol
TRD	Transit Routeing Domain

5. General Protocol Information

IDRP is a routeing information exchange protocol which is located within the Network layer and interfaces to ISO 8473, which serves as a SNICP (see Figure 3). In particular, BISPDUs are encapsulated as the data portion of ISO 8473 NPDUs. IDRP is a connection-oriented protocol which is implemented only in Intermediate systems. Routeing and control information is carried in BISPDUs (described in clause 6), which flow on connections between pairs of BISs. Each BISPDU is packaged within one or more NPDUs for transmission by the underlying Network service. IDRP relies on the underlying Network service to provide for fragmentation and reassembly of BISPDUs. IDRP queues Outbound BISPDUs as input to the underlying Network Layer service, retaining a copy of each BISPDU until an acknowledgement is received. Similarly, inbound BISPDUs are queued as input to the BISPDU-Receive process.

IDRP exchanges BISPDUs in a reliable fashion. It provides mechanisms for the ordered delivery of BISPDUs and for the detection and retransmission of lost or corrupted BISPDUs. The mechanisms for achieving reliable delivery of BISPDUs are described in clause 7.7; methods for establishing BIS-BIS connections are described in clause 7.6.

IDRP is consistent with the routeing model presented in ISO TR 9575. To emphasize its policy-based nature, the IDRP routeing model includes a Policy Information Base, as shown in Figure 4. IDRP can be described in terms of four major components:

a) **BISPDU-Receive Process:** responsible for accepting and processing control and routeing information from the local environment and from BISPDUs of other BISs. This information is used for a variety of purposes, such as receiving error reports and guaranteeing reliable reception of BISPDUs from neighboring BISs. (For example, the Update-Receive process (see 7.14) is the part of the BISPDU-Receive process that deals with the reception of routeing information after a BIS-BIS connection has been established.)

b) **BISPDU-Send Process:** responsible for constructing BISPDUs which contain control and routeing information. BISPDUs are used by the local BIS for a variety of purposes, such as advertising routeing information to other BISs, initiating BIS-BIS communication, and validating BIS routeing information bases.

c) **Decision Process:** responsible for calculating routes which will be consistent with local routeing policies. It operates on information in both the PIB and the Adj-RIBs, using it to create the Local RIBs (Loc-RIBs) and the local Forwarding Information Bases (see clause 7.10).

d) **Forwarding Process:** responsible for supplying resources to accomplish relaying of NPDUs to their destinations. It uses the FIB(s) created by the Decision Process.

5.1 Inter-RD Topology

This protocol views the overall global OSIE as an arbitrary interconnection of Transit Routeing Domains and End Routeing Domains which are connected by real inter-domain links placed between BISs located in the respective routeing domains. This standard provides for the direct exchange of routeing information between BISs, which may be located either in the same routeing domain or in adjacent routeing domains.

5.2 Routeing Policy

The direct exchange of policy information is outside the scope of IDRP. Instead, IDRP communicates policy information indirectly in its UPDATE PDUs which reflect the effects of the local policies of RDs on the path to the destination. Since all BISs within a routeing domain must enforce consistent active routeing policies, IDRP provides methods for detecting the existence of active inconsistent policies within a routeing domain. However, the semantics of routeing policies and the methods for establishing them are outside the scope of this standard.

NOTE 2: Annex K illustrates a policy description method and its associated semantics as one example of how policies might be expressed.

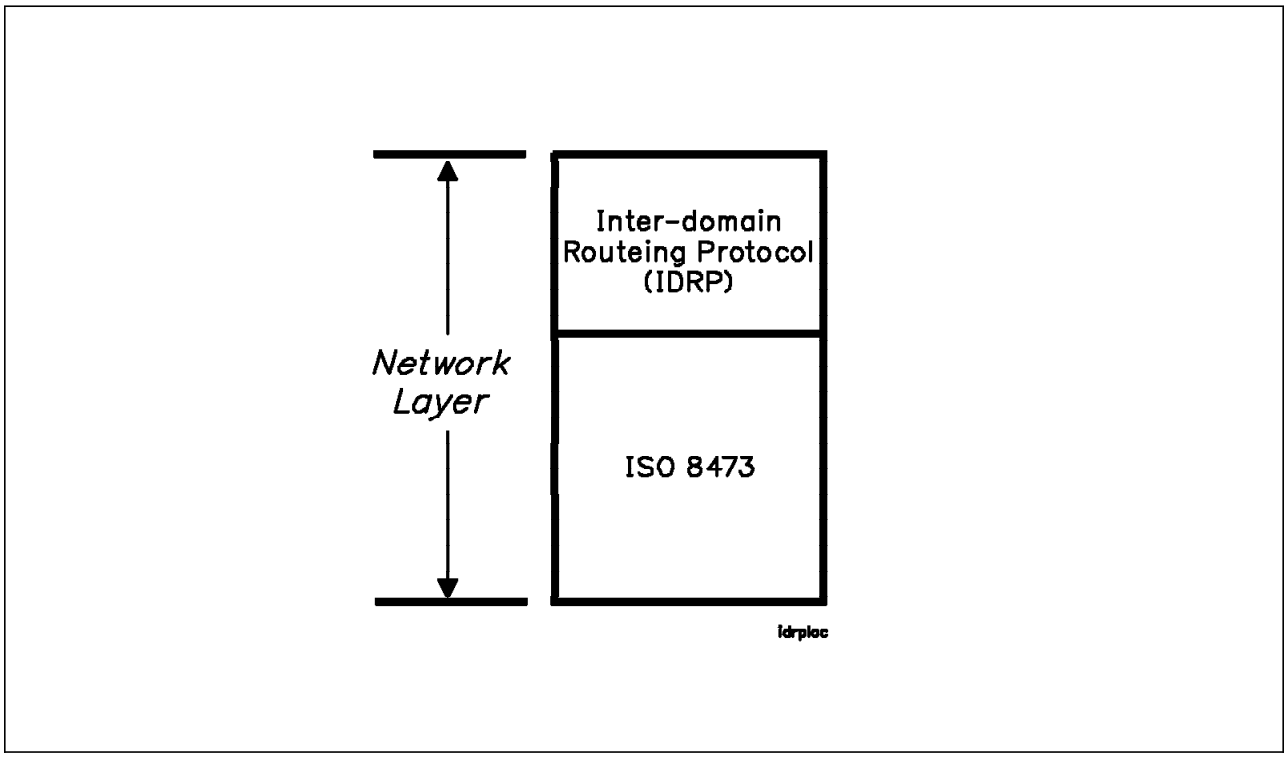

Figure 3. Position of IDRP within Network Layer

Each routeing domain chooses its routeing policies independently, and insures that all its BISs calculate inter-domain paths which satisfy those policies. Local routeing policies are applied to information in the Routeing Information Base (RIB) to determine a degree of preference for potential paths (see 7.16). From those paths which are not rejected by the routeing policy, a BIS selects the paths which it will use locally; from the locally selected paths, the BIS will then select the paths that it will advertise externally.

To enforce routeing policies and to insure that policies are both feasible and consistent, this protocol:

— carries path information, expressed in terms of Routeing Domain Identifiers (RDIs) and various path attributes, in its UPDATE PDUs

— permits a routeing domain to selectively propagate its reachability information to a limited set of other routeing domains

— provides a method to detect policy inconsistencies within the set of BISs located in a single routeing domain.

— permits each routeing domain to set its policies individually: that is, global coordination of policy is not required.

The set of rules that comprises the routing policy enforced by a BIS are held in a Policy Information Base (PIB), which is separate from the RIB. Depending on local Security and QOS requirements, the PIB may also contain:

a) rules for the aggregation of routes that include the SECURITY and LOCALLY DEFINED QOS path attributes (see clause 7.18.2)

b) rules for enforcing local QOS Maintenance Policies and the effective Security Policy, during NPDU forwarding

c) rules for updating SECURITY and LOCALLY DEFINED QOS path attributes in routes that are re-advertised to external routeing domains.

5.3 Types of Systems

An Intermediate system that implements the protocol described in this international standard is called a Boundary Intermediate system (BIS). Each BIS resides in a single routeing domain, and may optionally act simultaneously as a BIS and as an intra-domain IS within its own routeing domain.[3]

5.4 Types of Routeing Domains

The protocol described in this International Standard recognizes two types of routeing domains, end routeing domains and transit routeing domains; each of them may contain both ISs and ESs.

5.5 Routeing Domain Confederations

IDRP provides support for Routeing Domain Confederations (RDCs); this optional function permits groups of routeing domains to be organized in a hierarchical fashion.

An RDC is formed by means outside the scope of this protocol, and composed of a set of *confederation members*. Confederation members (CMs) are either individual routeing domains or routeing domain confederations. Thus, the definition of an RDC is recursive: a confederation member may be a single routeing domain or another confederation.

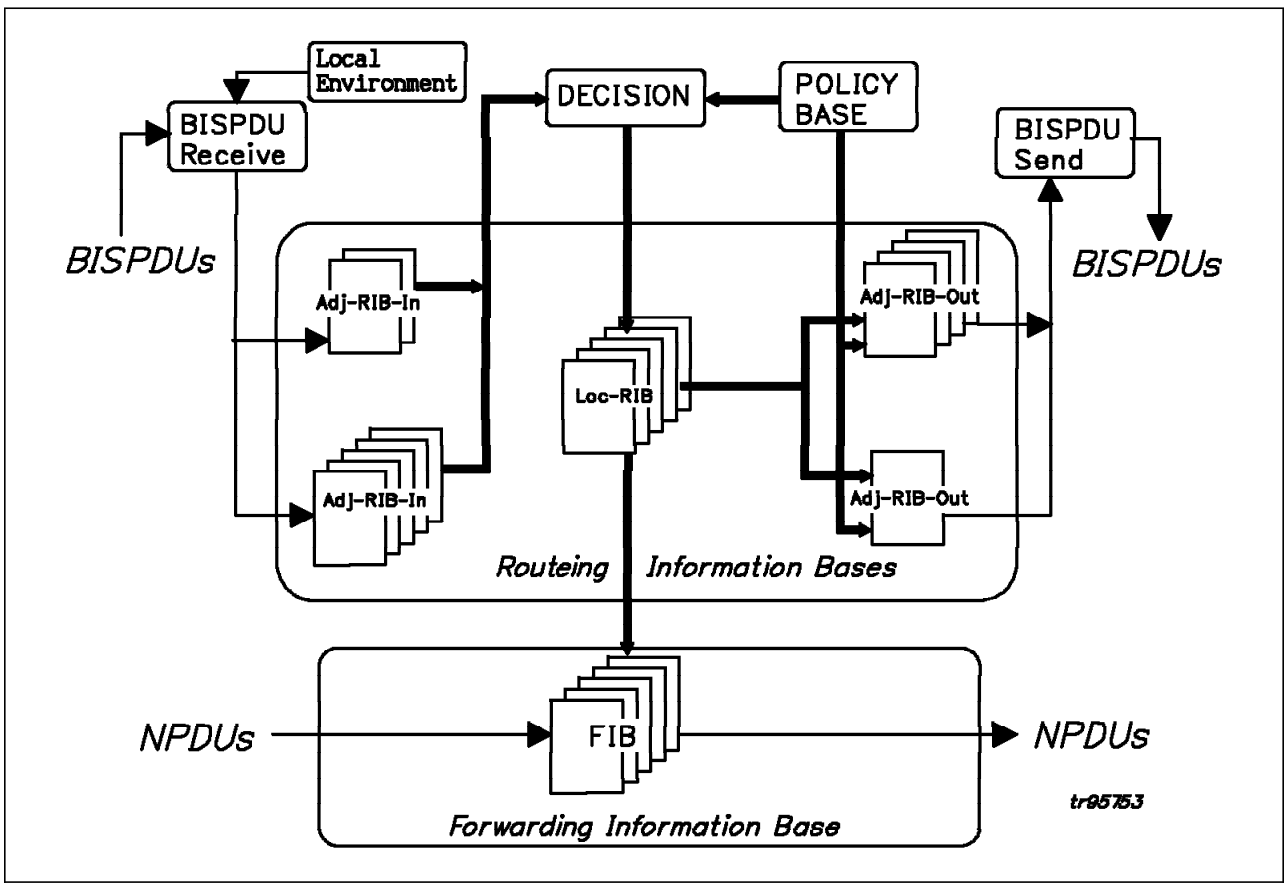

Figure 4. Inter-domain Routeing Components

[3] For example, a single system may simultaneously play the roles of a BIS for Inter-domain routeing and a level-2 IS for Intra-domain routeing as described in ISO/IEC 10589.

5.6 Routes: Advertisement and Storage

For purposes of this protocol, a *route* is defined as a unit of information that pairs destinations with the attributes of a path to those destinations:

— *Routes* are advertised between a pair of BISs in UPDATE PDUS: the *destinations* are the systems whose NSAP prefixes are reported in the NLRI field, and the *path* is the information reported in the path attributes fields of the same UPDATE PDU.

— *Routes* are stored in the Routeing Information Bases: namely, the Adj-RIBs-In, the Loc-RIBs, and the Adj-RIBs-Out. Routes that will be advertised to other BISs must be present in the Adj-RIBs-Out; routes that will be used by the local BIS must be present in the Loc-RIBs, and the next hop for each of these routes must present in the local BIS's Forwarding Information Bases; and routes that are received from other BISs are present in the Adj-RIBs-In.

— A *Route-ID* is assigned to each route that is advertised by a BIS. This identifier is unambiguous in the context of the BIS-BIS connection between the advertising BIS and the receiving BIS.

A BIS can support multiple routes to the same destination by maintaining multiple RIBs and the corresponding multiple FIBs. Each Loc-RIB will be identified by a different RIB-Att (see clauses 5.7 and 5.8); an Adj-RIB-Out shall contain at most one route to a particular destination.

If the BIS chooses to advertise the route, it may add to or modify the path attributes of the route before advertising it to adjacent BISs. For example, it is possible under certain circumstances to aggregate path attributes, NLRI, or entire routes, as described more fully in 7.18.2; or, as another example, the further distribution of a route may be restricted through the use of the DIST_LIST_EXCL attribute, as described in 7.12.6.

IDRP also provides mechanisms by which a BIS can inform its neighbor that a previously advertised route is no longer available for use. There are three methods by which a given BIS can indicate that a route has been withdrawn from service:

a) the Route-ID for a previously advertised route can be advertised in the WITHDRAWN ROUTES field of an UPDATE PDU, thus marking the associated route as being no longer available for use
b) a replacement route (with the same distinguishing attributes and NLRI) can be advertised, or
c) the BIS-BIS connection can be closed, which implicitly removes from service all routes which the pair of BISs had advertised to each other.

5.7 Distinguishing Path Attributes and RIB-Atts

Certain path attributes are classified as *Distinguishing Attributes*. Each distinct combination of such attributes identifies a particular information base which will be used to store information about the route. Each combination of distinguishing attributes is called a *RIB-Att* (RIB attribute); the RIB-Att is a common identifier for the Adj-RIB-In, Loc-RIB, Adj-RIB-Out, and FIB with which the route information is associated.

The number of RIB-Atts is limited by the number of distinct sets of permissible distinguishing attributes (see 7.11.2); this in turn limits the number of RIBs and FIBs that a BIS can support. The number of RIBs and FIBs can be further constrained by local decisions—a BIS may choose to support only a limited number of distinct routeing information bases (that is, a limited number of RIB-Atts, as described in clause 7.10.1).

5.8 Selecting the Information Bases

Each RIB is identified by a RIB-Att (RIB attribute), and the same RIB-Att also uniquely identifies the associated FIB.

When a BIS receives an UPDATE PDU, it determines the ROUTE_ID, LOCAL_PREF, and the set of distinguishing path attributes associated with each route that is advertised. The set of distinguishing path attributes contained between a pair of consecutively occurring ROUTE_SEPARATORs or between the last ROUTE_SEPARATOR and the end of the BISPDU unambiguously determine the RIB-Att for that route.

When a BIS receives an NPDU, it must unambiguously determine the FIB that should be used for forwarding this NPDU. This is accomplished by mapping certain fields in the header of the NPDU into a RIB-Att, which then unambiguously identifies a particular FIB (see clauses 8.2 and 8.3).

A general summary of IDRP's information bases is depicted in Table 1.

5.9 Routeing Information Exchange

This International Standard provides several rules governing the distribution and exchange of routeing information:

— rules for distributing routeing information internally (to BISs within a routeing domain)

— rules for distributing routeing information externally (to BISs in adjacent routeing domains)

Routeing information is carried in the protocol's BISPDUs, which are generated on an event-driven

Table 1. The IDRP Information Bases. The indexing variables and contents of the RIBs and FIBs are shown.

Information Base	*Indexed by...*	*Contains...*
Adj-RIB-In	— NET of adjacent BIS — RIB-Atts — Route-ID	— Path attributes — NLRI
Loc-RIB	— RIB-Atts	— Path attributes — NLRI
Adj-RIB-Out	— NET of adjacent BIS — RIB-Atts — Route-ID	— Path attributes — NLRI
FIB	— RIB-Atts — NLRI	— NET of next hop BIS — Output SNPA of local BIS — Input SNPA of next hop BIS — minimum priority associated with this subnetwork, when the RIB-Att contains the PRIORITY attribute — security related information associated with this subnetwork, when the RIB-Att contains the SECURITY attribute — QOS metric value, when the RIB-Att contains a RESIDUAL ERROR, TRANSIT DELAY, EXPENSE, or LOCALLY DEFINED QOS attribute

NOTES:

a) As a local option, a BIS may elect to apply information reduction techniques to path attributes and NLRI information.

b) For each adjacent BIS, a given BIS maintains an Adj-RIB-In for each RIB-Att (including the Empty RIB-Att) that it supports.

c) A BIS maintains a separate Loc-RIB for each RIB-Att (including the Empty RIB-Att) that it supports.

d) For each adjacent BIS, a given BIS maintains an Adj-RIB-Out for each set of RIB-Atts (including the Empty RIB-Att) that it advertises to that neighbor.

e) A given BIS maintains a separate FIB for each set of RIB-Atts (including the empty RIB-Att) that it supports— that is, each FIB corresponds to a Loc-RIB.

To facilitate the forwarding process, a BIS can organize each of its FIBS into two conceptual parts: one containing information for NLRI located within its own RD, and another for NLRI located in other RDs (see 8). For external NLRI, a BIS can further organize the FIB information based on whether the next-hop-BIS is located within its own RD or in another RD (see 8.4, items ″a″ and ″b″). And finally, for those next-hop BISs located in its own RD, the local BIS can organize the information according to a specific forwarding mechanism (see 8.4, items ″b1″ and ″b2″).

basis whenever a BIS receives information which causes it advertise new paths.

5.9.1 Internal Neighbor BIS

Each BIS establishes and maintains communications with all other BISs located in its routeing domain. The identity of all BISs within a routeing domain is contained in managed object **INTERNAL_BIS** described in clause 7.3.

5.9.2 External Neighbor BIS

Each BIS may establish and maintain communications with other BISs in adjacent routeing domains. A BIS has no direct communications link with any BIS in another routeing domain unless that RD is adjacent to it, as defined in clause 3.6: that is, a BIS does not communicate directly with a BIS located in a different routeing domain unless the pair of BISs are attached to at least one common subnetwork. The identity of neighbor BISs in adjacent routeing domains is contained in managed object **EXTERNAL-BIS-NEIGHBORS** described in clause 7.3.

NOTE 3: In the absence of an implementation specific method for ascertaining that a neighbor BIS listed in managed object **EXTERNAL-BIS-NEIGHBORS** is located on a common subnetwork with itself, a local

BIS can include the ISO 8473 Complete Route Record parameter so that the recipient of the BISPDU can determine whether the sending BIS is located on the same subnetwork as itself.

5.10 Routeing Domain Identifiers

Each routeing domain (RD) and routeing domain confederations (RDC) whose BIS(s) implement the protocol described in this International Standard shall have an unambiguous routeing domain identifier (RDI), which is a generic Network entity title, as described in ISO 7498.

An RDI is assigned statically in accordance with ISO 8348/Add.2, and does not change based on the operational status of the routeing domain. An RDI identifies a routeing domain or a confederation uniquely, but does not necessarily convey any information about its policies or the identities of its members.

5.11 Formats of RDIs, NETs, and NSAP Addresses

Within routeing domains whose BISs implement the protocol defined in this International Standard, RDIs, NETs and NSAP addresses shall be structured as described in clause 7.1.

All Boundary Intermediate systems shall be able to generate and forward PDUs containing NSAP addresses or NETs whose abstract syntax is as described in ISO 8348/AD2.

5.12 Design Objectives

The protocol described in this international standard was developed with a view towards satisfying certain design goals, while others specifically were not addressed by the mechanisms of this protocol.

5.12.1 Within the Scope of the Protocol

This International Standard supports the following design requirements:

Control Transit through an RD: It provides mechanisms to permit a given routeing domain to control the ability of NPDUs from other routeing domains to transit through itself.

Network Layer Protocol Compatibility: It does not require that any changes be made to the following existing Network layer protocols: ISO 8473, ISO 9542, ISO 10030, ISO 10589, and IS 8208.

Autonomous Operation: It provides stable operation in the global OSIE where significant sections of the interworking environment will be controlled by disjoint entities.

Distributed Information Bases: It does not require a centralized global repository for either routeing information or policy information.

QOS-based Routeing: It provides the ability to select routes based on the QOS characteristics of the NPDUs.

Deliverability: It accepts and delivers NPDUs addressed to reachable routeing domains and rejects NPDUs addressed to routeing domains known to be unreachable.

Adaptability: It adapts to topological changes between routeing domains, but not to traffic changes.

Promptness: It provides a period of adaptation to topological changes between domains that is a reasonable function of the maximum logical distance between any pair of routeing domains participating in an instance of this protocol.

Efficiency: It is efficient in the use of both processing resources and memory resources; it does not create excessive routeing traffic overhead.

Robustness: It recovers from transient errors such as lost or temporarily incorrect routeing PDUs, and it tolerates imprecise parameter settings.

Stability: It stabilizes in finite time to ″good routes″, as long as there are no continuous topological changes or corruptions of the routeing and policy information bases.

Heterogeneity: It is designed to operate correctly over a set of routeing domains that may employ diverse intra-domain routeing protocols. It is capable of running over a wide variety of subnetworks, including but not limited to: ISO 8208 and X.25 subnetworks, PSTN networks, and the OSI Data Link Service.

Availability: It will not result in inability to calculate acceptable inter-domain paths when a single point of failure happens for a pairing of topology and policy that have a *cut set* greater than one.

Fire walls: It will provide fire walls so that:

- Problems within one routeing domain will not affect intra-domain routeing in any other routeing domain
- Problems within one routeing domain will not affect inter-domain routeing, unless they occur on internal inter-domain Links
- Inter-domain routeing will not adversely affect intra-domain routeing.

Scaling: It imposes no upper limit on the number of routeing domains that can participate in a single instance of this protocol.

Multiple Routeing Administrations: It will accommodate inter-domain route calculations without regard to whether or not the participating routeing domains are under control of one or several administrative authorities.

5.12.2 Outside the Scope of the Protocol

The following requirements are not within the design scope of this protocol:

User Data Security: The security of user data carried within an NPDU is outside the scope of this protocol. This protocol is not designed to serve as a substitute for conventional data security practices. However, it can provide a security-related facility to the extent that route computation can be based upon the contents of the ISO 8473 security parameter.

Traffic Adaptation: It does not automatically modify routes based on the traffic load.

Guaranteed delivery: It does not guarantee delivery of all offered NPDUs.

Repair of Partitioned Routeing Domains: It does not use paths external to a partitioned routeing domain to repair the partition.

Repair of Partitioned Routeing Domain Confederations: It does not use paths external to a partitioned routeing domain confederation to repair the partition.

Shared Use of Inter-domain Links: It will not permit an external inter-domain link to carry intradomain traffic.

Suppression of Transient Loops: Although it provides mechanisms to detect and suppress looping of routeing information, it provides no mechanisms to detect or suppress transient looping of ISO 8473 NPDUs.

6. Structure of BISPDUs

In this document, the term *BISPDU* (Boundary IS PDU) is used as a general term to refer to any of the PDUs defined in this clause.

Octets in a PDU are numbered starting from 1, in increasing order. Bits in an octet are numbered from 1 to 8, where bit 1 is the low-order bit and is pictured on the right. When consecutive octets are used to represent a number, the lower octet number has the most significant value. The more significant semi-octet of each pair of semi-octets in a given octet is encoded in the high order bit positions (8 through 5).

Values are given in decimal, and all numeric fields are unsigned, unless otherwise noted.

The types of PDUs used by this protocol are:

— OPEN PDU
— UPDATE PDU
— IDRP ERROR PDU
— KEEPALIVE PDU
— CEASE PDU
— RIB REFRESH PDU

6.1 Header of BISPDU

Each BISPDU has a fixed size header. There may or may not be a data portion following the header, depending on the PDU type.

The layout of the header fields and their meanings are shown below:

Inter-domain Routeing Protocol Identifier (1 octet)
BISPDU Length (2 octets)
BISPDU Type (1 octet)
Sequence (4 octets)
Acknowledgement (4 octets)
Credit Offered (1 octet)
Credit Available (1 octet)
Validation Pattern (16 octets)

The meaning and use of these fields are as follows:

Inter-domain Routeing Protocol Identifier: An architectural constant for use in identifying the protocol by the methods of ISO DTR 9577.

BISPDU Length: The BISPDU Length field is a 2 octet unsigned integer. It contains the total length in octets of this BISPDU, including both header and data portions. The value of the BISPDU Length field shall be at least **MinBISPDULength** octets, and no greater than the value carried in the Maximum_PDU_Size field of the OPEN PDU received from the remote BIS.

Further, depending on the PDU type, there may be other constraints on the value of the Length field; for example, a KEEPALIVE PDU must have a length of exactly 30 octets. No padding after the end of BISPDU is allowed, so the value of the Length field must be the smallest value required given the rest of the BISPDU.

BISPDU Type: The BISPDU Type field contains a one octet type code which identifies the specific type of the PDU. The supported type codes are:

BISPDU Type	Code
OPEN PDU	1
UPDATE PDU	2
IDRP ERROR PDU	3
KEEPALIVE PDU	4
CEASE PDU	5
RIB REFRESH PDU	6

All other BISPDU type codes are reserved for future extensions.

Sequence: The Sequence field contains a 4 octet unsigned integer that is the sequence number of this PDU. The procedures for generating sequence numbers for the various types of BISPDU are described in clause 7.7.4.

Acknowledgement: The Acknowledgment field is a 4 octet unsigned integer that contains the sequence number of the PDU that the sender last received correctly and in sequence number order.

Credit Offered: The contents of this field indicate the number of additional BISPDUs that the sender is willing to accept from the remote BIS; it is used by the flow control process described in 7.7.5.

Credit Available: The contents of this field indicate the number of additional BISPDUs that the sender is able to send to the remote BIS; it is used by the flow control process described in 7.7.5.

Validation Pattern: This 16-octet field is used to provide a validation function for the BISPDU. Depending upon the contents of the field ″Authentication Code″ of the OPEN PDU, this field can provide:

- — data integrity for the contents of the BISPDU (see 7.7.1 and 7.7.3), or
- — data integrity for the contents of the BISPDU plus authentication of the peer BIS (see 7.7.2).

6.2 OPEN PDU

The OPEN PDU is used by a BIS for starting a BIS-BIS connection. The first PDU sent by either side is an OPEN PDU.

The OPEN PDU contains a fixed header and the additional fields shown below:

Fixed Header
Version (1 octet)
Hold Time (2 octets)
Maximum PDU Size (2 octets)
Source RDI Length Indicator (1 octet)
Source RDI (variable)
RIB-AttsSet (variable)
Confed-IDs (variable)
Authentication Code (1 octet)
Authentication Data (variable)

The meaning and use of these fields are as follows:

Version: This one octet field contains the version number of the protocol. Its value is currently 1.

Hold Time: This field contains a 2 octet unsigned integer that is the maximum number of seconds that may elapse between the receipt of two successive BISPDUs of any of the following types: KEEPALIVE, UPDATE, or RIB REFRESH PDUs.

Maximum PDU Size: This field contains a 2 octet unsigned integer that is the maximum number of octets that this BIS will accept in an incoming UPDATE PDU, IDRP ERROR PDU, or RIB REFRESH PDU.

Independent of this value, every BIS shall accept KEEPALIVE PDUs or CEASE PDUs of length 30 octets; every BIS shall also accept any OPEN PDU with length less than or equal to 3000 octets.

As a minimum, every BIS is required to support reception of all BISPDUs whose size is greater than or equal to **MinBISPDULength** octets and less than or equal to 1024 octets: that is, the minimum acceptable value for this field is 1024.

Source RDI Length Indicator: This one octet field contains the length in octets of the Source RDI field.

Source RDI: This variable length field contains the RDI of the routeing domain in which the BIS that is sending this BISPDU is located.

RIB-AttsSet: This variable length field contains a list of all *RIB-Att*s that the local BIS is willing to support when communicating with the neighbor BIS (that is, the BIS to which this OPEN PDU wis being sent).

It contains an encoding of all or part of the information contained in managed object **RIB-AttsSet** . (See clauses 7.3 and 7.10.1.) A BIS is not required to list all of its supported RIB-Atts in an OPEN PDU that is sent to a neighbor BIS located in an adjacent routeing domain. It must include only those RIB-Atts that correspond to Adj-RIBs-Out that the

local BIS will use to communicate with the neighbor BIS, and those that correspond to the RIB-Atts of the Adj-RIBs-In that the local BIS supports for storing UPDATE PDUs received from that neighbor BIS.

However, a BIS shall include all of the RIB-Atts listed in managed object **RIB-AttsSet** in an OPEN PDU that is sent to another BIS located in its own routeing domain. Failure to do so will result in an OPEN PDU error, as described in clause 7.20.2.

The encoding of this field is as follows:

Number of Non-empty RIB-Atts
Number of Distinguishing Attributes in First RIB-Att
First attribute, first RIB-Att
....
Last Attribute, first RIB-Att
Number of Distinguishing Attributes in Second RIB-Att
First attribute, second RIB-Att
....
Last Attribute, second RIB-Att
...
Number of Distinguishing Attributes in Nth RIB-Att
First attribute, Nth RIB-Att
....
Last Attribute, Nth RIB-Att
...
Number of Distinguishing Attributes in Last RIB-Att
First attribute, last RIB-Att
....
Last Attribute, last RIB-Att

The field *Number of RIB-Atts* is one octet long. It contains the total number of non-empty RIB-Atts supported by this BIS. Since every BIS supports an empty RIB-Att (see clause 7.10.1), the empty RIB-Att shall not be listed in the OPEN PDU.

If a BIS supports no RIB-Atts other than the empty RIB-Att, then the field *Number of Non-empty RIB-Atts* shall contain 0.

If the *Number of Non-Empty RIB-Atts* is non-zero, then the BIS supports all of the listed RIB-Atts plus the empty RIB-Att.

Each field *Number of Distinguishing Attributes in the Nth Set* is one octet long, and it contains the number of distinguishing attributes that are contained in the *Nth* RIB-Att. Since a RIB-Att consists of a set of distinguishing attributes, there is no significance to the order in which the distinguishing attributes of a given RIB-Att are listed.

Each of the individual attributes in a set is encoded as follows:

— a type specific distinguishing attribute (see clause 7.11.2) is encoded as a single octet, using the type code specified in clause 6.3 for the corresponding UPDATE PDU path attribute.

— a type-value-specific distinguishing attribute (see clause 7.11.2) is encoded as a triple <type, length, value>, using the type code for the attribute, the length in octets of the subsequent value, and the value itself. The value is encoded as follows for the following distinguishing attributes:

SECURITY: The value shall contain the Security Registration Identifier that identifies the Security Authority for which security related information is supported by this RIB-Att, encoded identically to the encoding of the SECURITY attribute specified in clause 6.3.1.14 including length fields, but with a zero length security information.

LOCALLY DEFINED QOS: The value shall comprise the NSAP Address Prefix of the QoS Authority and the QoS value that identifies the QoS measurement, encoded identically to the encoding of the LOCALLY DEFINED QOS attribute specified in clause 6.3.1.11 including length fields, but with a zero length metric.

Confed-IDs: This is a variable length field which reports the RDIs of all RDCs that this BIS is a member of. The encoding of this field is as follows:

Number of RDCs (1 octet)
Length of First RDI (1 octet)
RDI of first RDC
....
Length of Last RDI (1 octet)
RDI of last confederation

The 1 octet field *Number of RDCs* gives the number of RDCs of which this BIS is a member. A value of zero indicates that this BIS participates in no RDCs.

For each such confederation, the following fields give the length and RDI for each confederation,

where each RDI is encoded according to the preferred binary encoding of ISO 8348/Add.2.

Authentication Code: This 1-octet unsigned integer indicates the authentication mechanism being used:

a) Code 1 indicates that the Validation Pattern field in the header of each BISPDU contains an unencrypted checksum that provides data integrity for the contents of that BISPDU. Its use is described in clause 7.7.1 Its use is described in clause 7.7.1.

b) Code 2 indicates that the Validation Pattern field in the header of each BISPDU provides both peer-BIS authentication and data integrity for the contents of the BISPDU. The specific mechanism used to generate the validation pattern is mutually agreed to by the pair of BISs, but is not specified by this international standard. Its use is described in clause 7.7.2.

c) Code 3 indicates that the Validation Pattern field in the header of each BISPDU contains an unencrypted checksum covering the concatenation of the contents of the BISPDU with untransmitted password string(s). Its use is defined in 7.7.3.

Authentication Data: The form and meaning of this field is a variable-length field that depends on the Authentication Code, as described in clause D.1. The length of the authentication data field can be determined from the Length field of the BISPDU header.

6.3 UPDATE PDU

An UPDATE PDU is used to advertise feasible routes to a neighbor BIS, or to withdraw multiple unfeasible routes from service (see 5.6). An UPDATE PDU may simultaneously advertise multiple feasible routes and withdraw multiple unfeasible routes from service. The UPDATE PDU always includes the fixed header, Unfeasible Route Count field, and Total Path Length Attributes field; it can optionally contain the other fields shown in Figure 5:

— if routes are being explicitly withdrawn from service, then the UNFEASIBLE ROUTE COUNT field will be non-zero, and the WITHDRAWN ROUTES fields will be present

— if feasible routes are being advertised, then the TOTAL PATH ATTRIBUTES LENGTH field will be non-zero, and the PATH ATTRIBUTES and NLRI fields will be present.

An UPDATE PDU can advertise multiple routes; a route is described by several path attributes, each of which is encoded as a 4-tuple. All path attributes contained in a given UPDATE PDU apply to the destinations carried in the Network Layer Reachability Information field of the UPDATE PDU.

An UPDATE PDU can list multiple routes to be withdrawn from service. Each such route is identified by its Route-ID, which unambiguously identifies the route in the context of the BIS-BIS connection in which it had been previously been advertised.

An UPDATE PDU that is used only to withdraw routes from service (but not to advertise any feasible routes) will not include Path Attributes or NLRI. Conversely, if an UPDATE PDU does not withdraw any routes from service, the UNFEASIBLE ROUTE COUNT field will contain the value 0, and WITHDRAWN ROUTES field will not be present.

The components of the UPDATE PDU are described below:

Fixed Header
Unfeasible Route Count (2 octets)
Withdrawn Routes (variable)
Total Path Attributes Length (2 octets)
Path Attributes (variable)
Network Layer Reachability Information (variable)

The use of these fields is as follows:

Unfeasible Route Count: This is a 2-octet long field that contains an integer whose value is equal to the number of ROUTE-IDs that are included in the subsequent WITHDRAWN ROUTES field. A value of 0 indicates that no routes are being withdrawn from service, and that the WITHDRAWN ROUTES field is not present in this UPDATE PDU.

Withdrawn Routes: This is a variable length field that contains a list of Route-IDs for the routes that are being withdrawn from service. Each Route-ID is 4 octets in length.

Total Path Attribute Length: The length of this field is 2 octets. It is the total length of all Path Attributes in the UPDATE PDU in octets.

Path Attributes: A variable length sequence of path attributes is present in every UPDATE PDU that is used to advertise a feasible route.

Network Layer Reachability Information: A variable length field that list the destinations for the feasible routes that are being advertised in this UPDATE PDU.

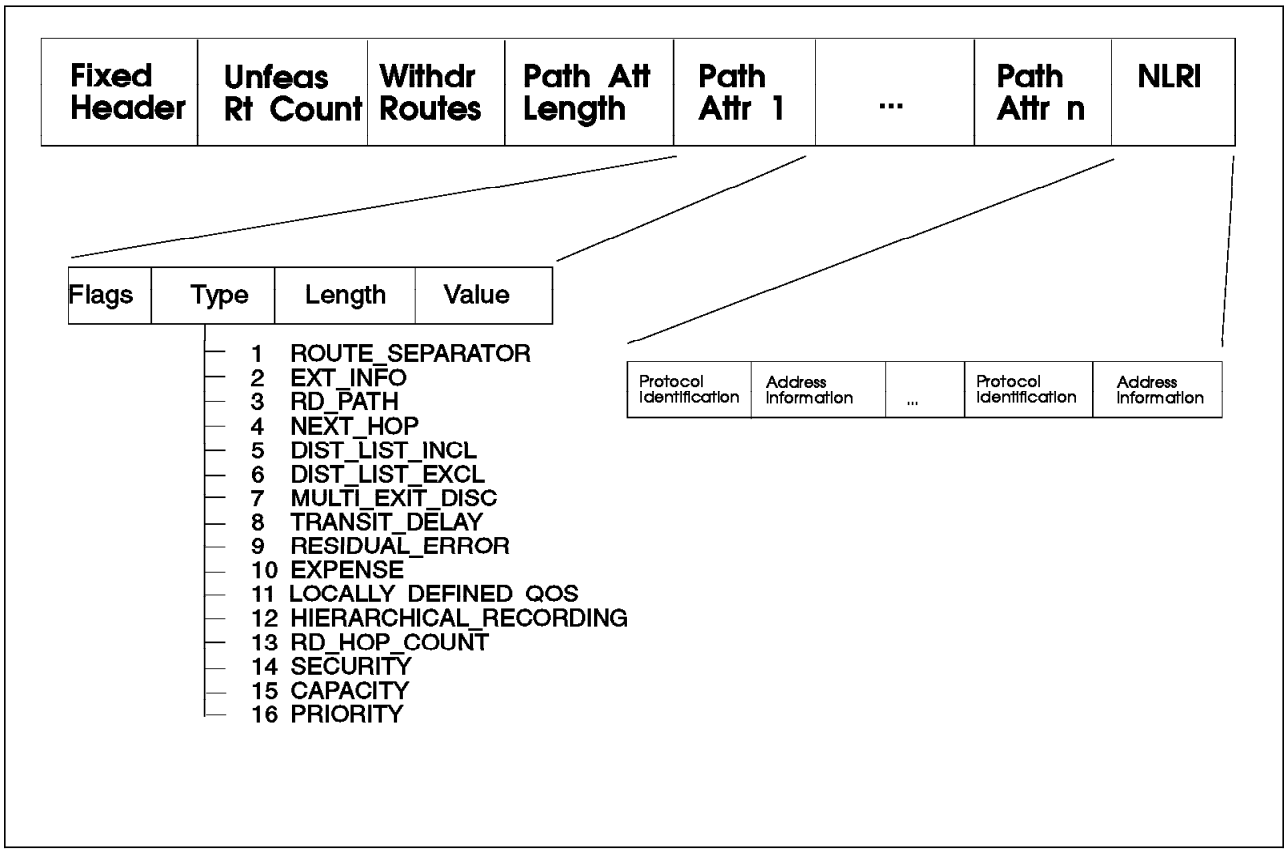

Figure 5. Structure of the UPDATE PDU

6.3.1 Path Attribute Encoding

Each path attribute is a 4-tuple of variable length—<flag, type, length, value>. The elements are used as follows:

— **Flag:**

The first element of each attribute is a one octet field:

- The high-order bit (bit 8) of the attribute flags octet is the Optional bit. If it is set to 1, the attribute is optional; if set to 0, the attribute is well-known.
- The second high-order bit (bit 7) of the attribute flags octet is the Transitive bit. It defines whether an optional attribute is transitive (if set to 1) or non-transitive (if set to 0). For well-known attributes, the Transitive bit shall be set to 1.
- The third high-order bit (bit 6) of the attribute flags octet is the Partial bit. It defines whether the optional transitive attribute is partial (if set to 1) or complete (if set to 0). For well-known attributes and for optional non-transitive attributes the Partial bit shall be set to 0.
- The lower order five bits (1 through 5) of the attribute flags octet are reserved. They shall be transmitted as 0 and shall be ignored when received.

— **Type:**

The second element of each attribute is a one octet field which contains the type code for the attribute. Currently defined attribute type codes are discussed in clause 7.11.

NOTE 4: It is not the intention of this international standard to define globally understood path attributes for type codes greater than value 128. Such codes are reserved for local use.

— **Length:**

The third field of each path attribute is 2 octets in length; it contains the length in octets of the immediately following Value field.

— **Value:**

The remaining octets of each path attribute field contain the value of the attribute, which is interpreted according to the attribute flags and the attribute type code. The supported attribute values and their encodings are defined below.

6.3.1.1 ROUTE_SEPARATOR (Type Code 1)

ROUTE_SEPARATOR is a well-known mandatory attribute whose length is 5 octets. One ROUTE_SEPARATOR attribute shall be present for each feasible route that is advertised in an UPDATE PDU. Multiple ROUTE_SEPARATORs may appear in a single UPDATE PDU. Each one provides a 2-tuple <ROUTE_ID, LOCAL_PREF> for the route whose distinguishing attributes immediately follow. It is encoded as shown below:

ROUTE-ID 1 (4 octets)
LOCAL-PREF 1 (1 octet)

a) ROUTE-ID:

A four octet field that contains the route identifier for the route associated with the RIB-Att composed of the set of distinguishing path attributes that appear between itself and either the next ROUTE_SEPARATOR attribute or the end of the UPDATE PDU.

b) LOCAL_PREF: A one octet field that contains the local preference value for route.

The usage of this attribute is defined in clause 7.12.1.

6.3.1.2 EXT_INFO (Type Code 2)

EXT_INFO is a well-known discretionary attribute that has a length of zero octets; its presence indicates that some (or all) of the path attributes or Network Layer Reachability Information contained in this UPDATE PDU have been obtained by methods not specified by IDRP. Conversely, its absence indicates that all path attributes and NLRI have been learned by methods defined within IDRP. Its usage is defined in clause 7.12.2

6.3.1.3 RD_PATH (Type Code 3)

The RD_PATH attribute is a well-known mandatory attribute composed of a series of RD path segments. Each path segment is represented by a triple <path segment type, path segment length, path segment value>.

The path segment type is a 1-octet long field, with the following values defined:

Segment Type	Value
RD_SET	1
RD_SEQ	2
ENTRY_SEQ	3
ENTRY_SET	4

An RD_SEQ and a ENTRY_SEQ provide a list of RDIs, for routeing domains or for confederations respectively, in the order that the routeing information has travelled through them. An RD_SET and an ENTRY_SET provide an unordered list of RDIs, for routeing domains or for confederations respectively; the routeing information has not necessarily travelled through all of the listed domains or confederations.

The path segment length is a two octet field containing the length in octets of the path segment value field.

The path segment value field contains one or more 2-tuples <length, RDI>. *Length* is a one octet long field that contains the length of the RDI in octets; *RDI* itself is encoded according to the preferred binary encoding of ISO 8348/Add.2.

Usage of this attribute is defined in clause 7.12.3.

6.3.1.4 NEXT_HOP (Type Code 4)

This is a well-known discretionary attribute that can be used for two principal purposes:

a) to permit a BIS to advertise a different BIS's NET in the "NET of Next Hop" field

b) to allow a given BIS to report some or all of the SNPAs that exist within the local system

It is encoded as shown below:

IDRP_Server_Allowed (1 octet)

followed by one or more of the following:

Proto_type (1 octet)
Proto_length (1 octet)
Protocol (variable)
Length of NET (1 octet)
NET of Next Hop (variable)
Number of SNPAs (1 octet)
Length of 1st SNPA(1 octet)
First SNPA (variable)
Length of 2nd SNPA (1 octet)
Second SNPA (variable)
...
Length of Last SNPA (1 octet)
Last SNPA (variable)

The use and meaning of these fields are as follows:

IDRP_Server_Allowed: The value X′FF′ indicates the recipient of this UPDATE PDU has the option of advertising (in its own outbound UPDATE PDUs) the NET and SNPA information learned from this UPDATE PDU. If the value is not X′FF′, then the recipient of this UPDATE PDU shall not advertise the NET and SNPA information learned from this UPDATE PDU.

Proto_type: This field indicates the type of protocol identifier that follows:

1 ISO TR 9577 IPI/SPI
2 ISO 8802 LSAP

For use with ISO 8473, this field shall contain the value 1.

Proto_length: This field indicates the length in octets of the protocol identifier that follows. For use with ISO 8473, this field shall contain the value 1.

Protocol: This field carries the identity of the protocol associated with the address information that follows. For use with ISO 8473, this field shall contain the value X′81′.

Length of NET: A 1 octet field whose value expresses the length of the ″NET of Next Hop″ field as measured in octets

NET of Next Hop: A variable length field that contains the NET of the next BIS on the path to the destination system

Number of SNPAs: A 1 octet field which contains the number of distinct SNPAs to be listed in the following fields. The value 0 may be used to indicate that no SNPAs are listed in this attribute.

Length of Nth SNPA: A 1 octet field whose value expresses the length of the ″Nth SNPA of Next Hop″ field as measured in semi-octets

Nth SNPA of Next Hop: A variable length field that contains an SNPA of the BIS whose NET is contained in the ″NET of Next Hop″ field. The field length is an integral number of octets in length, namely the rounded-up integer value of one half the SNPA length expressed in semi-octets; if the SNPA has an contains an odd number of semi-octets, a value in this field will be padded with a trailing all-zero semi-octet.

Its usage is defined in clause 7.12.4. A conforming IDRP implementation may ignore next hop information for all protocols other than ISO 8473. The Decision and Forwarding processes of IDRP for use with protocols other than ISO 8473 are outside the scope of this international standard.

NOTE 5: The ISO 8802 LSAP format is intended for use with protocols that are identified directly by the use of a particular LSAP value; in such cases, the value of the Proto_length field should be 1.

The ISO TR 9577 IPI/SPI format may include additional information beyond the IPI/SPI value in the Protocol field; for example, if the IPI/SPI value is X′80′, an IEEE 802.1a SNAP header might be included.

6.3.1.5 DIST_LIST_INCL (Type Code 5)

The DIST_LIST_INCL is a well-known discretionary attribute that contains a list of the RDIs of routeing domains and confederations to which the Network Layer Reachability Information contained in that UPDATE PDU may be distributed. Its usage is described in clause 7.12.5.

Each RDI in the list is encoded as a <length, value> pair, using the following fields:

Count (1 octet)
Length (1 octet)
Value (variable)
...
Length (1 octet)
Value (variable)

The use and meaning of these fields are as follows:

a) Count:

The count field is an integer that gives the number of RDIs in the list, where each RDI is represented by a <length, value> pair as described below.

b) Length:

The length field indicates the length in octets of the following RDI.

c) Value:

The value field contains the RDI, encoded according to the preferred binary encoding of ISO 8348/Add.2.

The DIST_LIST_INCL attribute shall not be present together with the DIST_LIST_EXCL attribute in the same UPDATE PDU.

6.3.1.6 DIST_LIST_EXCL (Type Code 6)

The DIST_LIST_EXCL is a well-known discretionary attribute that contains a list of the RDIs of routeing domains and confederations to which the Network Layer Reachability Information contained in that UPDATE PDU shall not be distributed. Its usage is described in clause 7.12.6.

Each RDI in the list is encoded as a <length, value> pair, using the following fields:

Count (1 octet)
Length (1 octet)
Value (variable)
...
Length (1 octet)
Value (variable)

The use and meaning of these fields are as follows:

a) Count:

The count field is an integer that gives the number of RDIs in the list, where each RDI is represented by a <length, value> pair as described below.

b) Length:

The length field indicates the length in octets of the following RDI.

c) Value:

The value field contains the RDI, encoded according to the preferred binary encoding of ISO 8348/Add.2.

The DIST_LIST_EXCL attribute shall not be present together with the DIST_LIST_INCL attribute in the same UPDATE PDU.

6.3.1.7 MULTI-EXIT_DISC (Type Code 7)

MULTI-EXIT_DISC (Multi-exit Discriminator) is an optional non-transitive attribute that is a 1 octet non-negative integer. The value of this attribute may be used by a BIS's decision process to discriminate between multiple exit points to an adjacent routeing domain. Its usage is defined in clause 7.12.7.

6.3.1.8 TRANSIT DELAY (Type Code 8)

The TRANSIT DELAY is a well-known discretionary attribute that has a length of 2 octets, and is used to notify a BIS that the path to destination has transit delay determined by the value of the attribute. Its usage is defined in 7.12.8.

6.3.1.9 RESIDUAL ERROR (Type Code 9)

The RESIDUAL ERROR is a well-known discretionary attribute that has a length of 4 octets, and is used to notify a BIS that the path to destination has residual error probability determined by the value of the attribute. Its usage is defined in clause 7.12.9.

6.3.1.10 EXPENSE (Type Code 10)

The EXPENSE is a well-known discretionary attribute that has a length of 2 octets, and is used to notify a BIS that the path to destination has expense determined by the value of the attribute. Its usage is defined in clause 7.12.10.

6.3.1.11 LOCALLY DEFINED QOS (Type Code 11)

This is a well-known discretionary attribute whose variable length field contains the parameters associated with a QOS related path attribute defined by a QOS authority.

The parameters of this attribute identify the QOS authority, the name of the QOS measurement, and, optionally, a metric associated with that measurement. The syntax and semantics of the QOS measurement and its metric are outside the scope of this international standard, and are specified by the responsible QOS Authority.

It is encoded as follows:

NSAP prefix length (1 octet)
NSAP prefix (variable)
QOS length (1 octet)
QOS value (variable)
Metric length (1 octet)
Metric value (variable)

The use and meaning of the fields is as follows:

a) NSAP prefix length:

The length field indicates the length in bits of the NSAP address prefix (see 7.1.2.1). A length of zero indicates a prefix that matches all NSAPs.

b) NSAP prefix:

If the Length field specifies an integral number of octets, then the Prefix field shall contain only the NSAP address prefix encoded according to 7.1.2.1. If the Length field does not specify an integral number of octets, then the Prefix field shall contain the NSAP address prefix encoded according to 7.1.2.1, followed by enough trailing zeroes to make the end of the field fall on an octet boundary.

This field identifies the QoS Authority and comprises an NSAP Address Prefix when the QoS Authority is also an NSAP Addressing Authority (see ISO 8473 clauses 7.5.6.1 and 7.5.6.2); the NSAP Address Prefix is that of the QoS Authority's Addressing Subdomain. When the QoS Authority is not also an NSAP Addressing

Authority, then this field contains an NET that uniquely identifies the QoS Authority.

c) QOS length:

Gives the length in octets of the QOS value field.

d) QOS value:

Contains the value of the QOS parameter.

e) Metric length:

Gives the length in octets of the metric value field. A length of zero is a permitted value.

f) Metric value:

Contains a positive integer that is the value of the metric associated with the path being advertised (that is, its "cost")

Its usage is defined in clause 7.12.11.

6.3.1.12 HIERARCHICAL RECORDING (Type Code 12)

This is a well-known discretionary attribute. It contains a 1-octet field used by members of a routeing domain confederation to control the transitivity of NPDUs through the confederation. It is encoded as follows:

Flag (1 octet)

Its usage is defined in clause 7.12.12.

6.3.1.13 RD_HOP_COUNT (Type Code 13)

The RD_HOP_COUNT is a well-known mandatory attribute that contains a 1 octet long field. It contains an unsigned integer that is the upper bound on the number of routeing domains through which this UPDATE PDU has travelled. Its usage is defined in clause 7.12.13.

6.3.1.14 SECURITY (Type Code 14)

This is a well-known discretionary attribute whose variable length field contains the parameters associated with a security related path attribute defined by a Security Authority.

The parameters of this attribute identify the Security Authority, and can also contain additional security related information, which may identify the protection provided by the route, a set of Security Labels associated with the route, or both. The syntax and semantics of the security related information are outside the scope of this international standard, and are specified by the responsible Security Authority.

The encoding of the attribute is as follows:

NSAP prefix length (1 octet)
NSAP prefix (variable)
Security Registration ID Length (1 octet)
Security Registration ID (variable)
Security Information Length (1 octet)
Security Information (variable)

The use and meaning of the fields is as follows:

a) NSAP Address Prefix Length:

The length field indicates the length in bits of the NSAP Address Prefix (see 7.1.2.1) encoded in the following field. This field may be set to zero if the Security Authority is identified by a Security Registration Identifier, and no NSAP Address Prefix is provided.

b) NSAP Address Prefix:

This field identifies the Security Authority and comprises an NSAP Address Prefix when the Security Authority is also an NSAP Addressing Authority (see ISO 8473 clauses 7.5.3.1 and 7.5.3.2); the NSAP Address Prefix is that of the Security Authority's Addressing Subdomain. When the Security Authority is not also an NSAP Addressing Authority, then this field contains an NET that uniquely identifies the Security Authority.

If the Length field specifies an integral number of octets, then the Prefix field shall contain only the NSAP address prefix encoded according to 7.1.2.1. If the Length field does not specify an integral number of octets, then the Prefix field shall contain the NSAP address prefix encoded according to 7.1.2.1, followed by enough trailing zeroes to make the end of the field fall on an octet boundary.

c) Security Registration ID Length:

This field contains the length in octets of the Security Authority's Security Registration Identifier. This field may be set to zero if the Security Authority is identified by an NSAP Address Prefix, and no Security Registration ID is provided. When both an NSAP Address Prefix and a Security Registration ID are present then they shall identify the same Security Authority.

d) Security Registration ID:

This variable length field contains the Security Registration Identifier of the Security Authority responsible for defining the following security information.

e) Security Information Length:

This field contains the length in octets of the Security Information, as a non-zero unsigned integer. A value of zero is not permitted.

f) Security Information:

This variable length field contains an integral number of octets comprising security related information specified by the Security Authority identified above. The syntax and semantics of this information are outside of the scope of this international standard.

6.3.1.15 CAPACITY (Type code 15)

CAPACITY is a well-known mandatory attribute that has a length of 1 octet, and is used to denote the relative capacity of the RD_PATH for handling traffic. High values indicate a lower traffic handling capacity than do low values. Its usage is defined in clause 7.12.15.

6.3.1.16 PRIORITY (Type Code 16)

PRIORITY is a well-known discretionary attribute that is 1 octet in length. The content of the field is an integer in the range from 0 to 14. It enables paths to be distinguished on the basis of which values of the ISO 8473 priority parameter (see ISO 8473, clause 7.5.7). As in ISO 8473, the value 0 indicates the normal (default) priority, and the values 1 through 14 indicate higher priorities. A particular value of this parameter, *m*, means that the BIS will not forward any ISO 8473 PDUs whose priority parameter has a value less than *m*. Detailed usage rules are presented in clause 7.12.16.

6.3.2 Network Layer Reachability Information

This variable length field contains a list of reachable destinations encoded as zero or more 5-tuples of the form <Proto_type, Proto_length, Protocol, Addr_length, Addr_info>, whose fields are described below:

Proto_type (1 octet)
Proto_length (1 octet)
Protocol (variable)
Addr_length (2 octets)
Addr_info (variable)

The use and meaning of these fields are as follows:

Proto_type: This field indicates the type of protocol identifier that follows:

1 ISO TR 9577 IPI/SPI

2 ISO 8802 LSAP

For use with ISO 8473, this field shall contain the value 1.

Proto_length: This field indicates the length in octets of the protocol identifier that follows. For use with ISO 8473, this field shall contain the value 1.

Protocol: This field carries the identity of the protocol associated with the address information that follows. For use with ISO 8473, this field shall contain the value X′81′.

Addr_length: This field specifies the total length in octets of the address information that follows.

Addr_info: This field contains a list of reachable address prefixes. The encoding of this field is specific to each protocol supported.

For use with ISO 8473, this field shall be encoded as one or more 2-tuples of the form <length, prefix>, whose fields are described below:

a) The Length field is one octet long, and it indicates the length in bits of the address prefix. A length of zero indicates a prefix that matches all addresses.

b) The Prefix field has a variable length, and it contains an address prefix. If the Length field does not specify an integral number of octets, then the Prefix field shall be padded with trailing zero bits to the next octet boundary. For purposes of use with ISO 8473, this field shall contain an NSAP address prefix encoded according to 7.1.2.1.

A conforming IDRP implementation may ignore NLRI for all protocols other than ISO 8473. The Decision and Forwarding processes of IDRP for use with protocols other than ISO 8473 are outside the scope of this international standard.

NOTE 6: The ISO 8802 LSAP format is intended for use with protocols that are identified directly by the use of a particular LSAP value; in such cases, the value of the proto_length field should be 1.

The ISO TR 9577 IPI/SPI format may include additional information beyond the IPI/SPI value in the Protocol field; for example, if the IPI/SPI value is X′80′, an IEEE 802.1a SNAP header might be included.

6.4 IDRP ERROR PDU

IDRP ERROR PDUs report error conditions which have been detected by the local BIS. In addition to its fixed header, the IDRP ERROR PDU contains the following fields:

Fixed Header
Error Code (1 octet)
Error Subcode (1 octet)
Data (variable)

The use of these fields is as follows:

Error code: The Error code field is 1 octet long, and shall be present in every IDRP ERROR PDU. It describes the type of error. The following error codes are defined:

Error Code	Value
OPEN_PDU_Error	1
UPDATE_PDU_Error	2
Hold_Timer_Expired	3
FSM_Error	4
RIB_REFRESH_PDU_Error	5

All errors are fatal to the BIS connection.

Error subcode: The Error subcode is one octet long, and shall be present in every IDRP ERROR PDU. The error subcode provides more specific information about the nature of the reported error. A given IDRP ERROR PDU may report only one error subcode for the indicated error code. The supported error subcodes are as follows:

a) OPEN PDU Error subcodes:

Subcode	Value
Unsupported_Version_Number	1
Bad_Max_PDU_Size	2
Bad_Peer_RD	3
Unsupported_Authentication_code	4
Authentication_Failure	5
Bad_RIB-AttsSet	6
RDC_Mismatch	7

b) UPDATE PDU Error subcodes:

Subcode	Value
Malformed_Attribute_List	1
Unrecognized_Well-known_Attribute	2
Missing_Well-known_Attribute	3
Attribute_Flags_Error	4
Attribute_Length_Error	5
RD_Routeing_Loop	6
Invalid_NEXT_HOP_Attribute	7
Optional_Attribute_error	8
Invalid_Reachability_Information	9
Misconfigured_RDCs	10
Malformed_NLRI	11
Duplicated_Attributes	12
Illegal_RD_Path_Segment	13

c) Hold_Timer_Expired Error Subcodes:

Subcode	Value
NULL	0

d) FSM_Error Error Subcodes:

When an FSM Error (see 7.6.1) has occurred, the first semi-octet of the error subcode carries the type number of the BISPDU that should not have been received and the second semi-octet encodes the state that the FSM was in when the reception took place. The BISPDU type numbers are defined in 6.1; the FSM states are encoded as follows:

FSM State	Encoded Value
CLOSED	1
OPEN-RECVD	2
OPEN-SENT	3
CLOSE-WAIT	4
ESTABLISHED	5

e) RIB_REFRESH_PDU_Error Subcodes:

Subcode	Value
Invalid_OpCode	1
Unsupported_RIB-Atts	2

Data: This variable length field contains zero or more octets of data to be used in diagnosing the reason for the IDRP ERROR PDU. The contents of the Data field depends upon the error code and error subcode.

Note that the length of the Data field can be determined from the Length field of the BISPDU header. The minimum length of the IDRP ERROR PDU is 32 octets, including BISPDU header.

6.5 KEEPALIVE PDU

A KEEPALIVE PDU consists of only a PDU header and has a length of 30 octets.

A BIS can use the periodic exchange of KEEPALIVE PDUs with an adjacent BIS to verify that the peer BIS is reachable and active. KEEPALIVE PDUs are exchanged often enough as not to cause the hold time advertised in the OPEN PDU to expire. The maximum time between KEEPALIVE PDUs shall be one third of the Hold Time interval as advertised in the Hold Time field of the OPEN PDU.

A KEEPALIVE PDU may be sent asynchronously to acknowledge the receipt of other BISPDUs. Sending a KEEPALIVE PDU does not cause the sender's sequence number to be incremented.

6.6 CEASE PDU

A CEASE PDU consists of only a PDU header and has length of 30 octets.

A CEASE PDU is used by the originating BIS to instruct the peer BIS to close the BIS-BIS connection.

Receipt of a CEASE PDU will cause the BIS to close down the connection with the BIS that issued it, as described in 7.6.2.

6.7 RIB REFRESH PDU

The RIB REFRESH PDU is used to allow a BIS to send a refresh of the routeing information in an Adj-RIB-Out to a neighbor BIS, or to solicit a neighbor BIS to send a refresh of its Adj-RIB-Out to the local BIS. The RIB REFRESH PDU contains a fixed header and also the additional fields shown below:

Fixed Header
OpCode (1 octet)
RIB-Atts (variable)

The use and meaning of these fields is as follows:

There are three OpCode values defined:

Code	Meaning
1	RIB Refresh Request
2	RIB Refresh Start
3	RIB Refresh End

The RIB-Atts field contains the RIB-Atts of the Adj-RIB-In for which a refresh is being requested. This field is encoded in the same way that the RIB-AttsSet field of the OPEN PDU is encoded.

Its usage is defined in clause 7.10.3.

7. Elements of Procedure

This clause explains the elements of procedure used by the protocol specified in this International Standard; it also describes the naming conventions and system deployment practices assumed by this protocol.

7.1 Naming and Addressing Conventions

Correct operation of this protocol requires that all NSAP-addresses, NETs, and RDIs shall be encoded according to the preferred binary encoding method of ISO 8348/Add.2.

7.1.1 Interpretation of Address Information

IDRP does not assume or require any particular internal structure for the address. That is, as long as the routeing domain administrator assigns values within this field that are consistent with the deployment constraints of clause 7.2.2, the protocol specified in this International Standard will operate correctly.

7.1.2 NSAP Address Prefixes

NSAP address prefixes provide a compact method for identifying groups of systems that reside in a given routeing domain or confederation. A prefix may have a length that is either smaller than, or the same size as, the base NSAP address.

NOTE 7: At one extreme, for example, the largest NSAP address prefix will be identical with the full NSAP address—in this case, it would identify a single system rather than a group of systems. At another extreme, the NSAP prefix may be based on only a portion of NSAP address's IDP—in this case, it will identify a large group of systems.

7.1.2.1 Encoding of Prefixes

The encoded form of an NSAP address prefix is obtained by encoding the prefix (expressed in its abstract syntax) according to the preferred binary encoding of ISO 8348/Add.2, unless the end of the prefix falls within the IDP. In this case, each decimal digit in the prefix shall be encoded as the corresponding semi-octet in the range 0000-1001 and no padding characters shall be inserted.

The length of an encoded prefix is denominated in bits. The prefix shall end on a boundary that is legal in the abstract syntax of the address family from which it is derived. For example, the encoding of a prefix whose DSP is expressed in decimal syntax must end on a semi-octet boundary, while the encoding of a prefix whose DSP is expressed in binary syntax can end on an arbitrary bit boundary.

7.1.2.2 Prefix Matching

As part of the forwarding process, a BIS must match the destination NSAP address from an ISO 8473 NPDU against the NSAP address prefixes that are used to identify the Forwarding Information Bases. An NSAP address prefix that extends into the DSP shall be compared directly against the encoded NSAP address, including any padding characters that may be

present. An NSAP address prefix which does not extend into the DSP shall be compared against the derived quantity *NSAP′*, which is obtained from the encoded NSAP address by removing all padding characters that were inserted by the binary encoding process of ISO 8348/Add.2

The existence of a match shall be determined as follows:

a) If the encoded NSAP address (or *NSAP′*) contains fewer bits than the NSAP address prefix, then there is no match.

b) If the encoded NSAP address (or *NSAP′*) contains at least as many bits as the NSAP address prefix, and all bits of the NSAP address prefix are identical to the corresponding leading bits of the encoded NSAP address (or *NSAP′*), there is a match. Otherwise, there is no match.

In cases where a given NSAP address matches several address prefixes, the match with the longest prefix shall take precedence. For purposes of prefix matching, the length of a prefix is defined to be the number of bits that it contains.

NOTE 8: Any implementation of a matching process that satisfies the requirements listed above may be used. The key point is that matching process must be aware of whether or not the NSAP address prefix extends into the DSP, and must then either include or exclude padding characters from the encoded NSAP address, as defined above.

7.2 Deployment Guidelines

7.2.1 Minimum Configuration of an RD

To participate in this inter-domain routeing protocol, a routeing domain must, as a minimum:

— contain at least one Boundary Intermediate system

— contain at least one BIS capable of delivering NPDUs to the intra-domain routeing function, if the RD contains End systems.

7.2.2 Deployment of ISs and ESs

End systems and intermediate systems may use any NSAP address or NET that has been assigned under ISO 8348/AD2 guidelines. However, correct and efficient operation of this protocol can only be guaranteed if the following additional guidelines are met:

a) An NSAP prefix carried in the NLRI field of route originated by a BIS in a given routeing domain should be associated with only that routeing domain: that is, no system identified by the prefix should reside in a different routeing domain. Ambiguous routeing may result if several routeing domains originate routes whose NLRI fields contain identical NSAP address prefixes, since this would imply that the same system(s) are simultaneously located in several routeing domains. (As noted in clause 7.1.2.2, prefixes may be discriminated by both content and length.)

b) Several different NSAP prefixes may be associated with a single routeing domain identifier (RDI). Thus, it is possible to construct a single routeing domain which contains a mix of systems which use NSAP addresses assigned by several different naming authorities.

The protocol defined in this international standard assumes that the guidelines listed above have been satisfied, but it contains no means to verify that this is so. Therefore, such verification will be the responsibility of the administrators of routeing domains.

7.3 Domain Configuration Information

Correct operation of the protocol described in this international standard assumes that a minimum amount of information is available to both the inter-domain and the intra-domain routeing protocols. The information is static in nature, and is not expected to change frequently. This protocol specifies the content of the information in terms of managed objects.

The information required by a BIS that implements the protocol described in this international standard is:

a) **Location and identity of adjacent intra-domain ISs:**

The managed object **INTRA-IS** lists the NETs of systems to which the local BIS may deliver an inbound NPDU whose destination lies within the BIS′s routeing domain. The managed object contains the NETs of ISs that support the intra-domain routeing protocol and are located on the same common subnetwork as the local BIS. In particular, if the BIS participates in the intra-domain routeing protocol (that is, the protocol machines for both intra- and inter-domain routeing are located in the same real system), then the NET of the local BIS will be listed in the managed object **INTRA-IS**.

b) **Location and identity of BISs in the domain:**

This information permits a BIS to identify all other BISs located within its routeing domain. This information is contained in the managed object **INTERNAL-BIS**, which contains a set of NETs which identify the BISs in the domain.

c) **Location and identity of BISs in adjacent domain:**

Each BIS needs information to identify the NETs of each BIS located in an adjacent RD and reach-

able via a single subnetwork hop. This information is contained in managed object **EXTERNAL-BIS-NEIGHBORS**, which consists of a list of NETs.

d) **NETs and NSAP addresses of all systems in the routeing domain:**

This information is used by the BIS to construct its network layer reachability information. This information is contained within the managed object **INTERNAL-SYSTEMS**, which lists the address prefixes of the systems contained within the routeing domain.

e) **Local RDI:**

This information is contained in managed object **LOCAL-RDI**; it is the RDI of the routeing domain in which the BIS is located.

f) **RIB-AttsSet:**

This managed object lists all of the RIB-Atts (RIB attributes) which are supported by BISs located in this routeing domain. As noted in clause 7.10.1, a RIB-Att is a set of Distinguishing Attributes.

g) **RDC-Config:**

This information identifies all the routeing domain confederations (RDCs) to which the RD of the local BIS belongs, and it describes the nesting relationships that are in force between them. It is contained in managed object **RDC-Config**.

h) **Local SNPAs:**

This managed object contains a list of the SNPAs of the BIS.

7.4 Advertising NLRI Information

The NLRI field in an UPDATE PDU contains information about the NSAP addresses of systems that reside within a given routeing domain or whose NSAP addresses are under control of the administrator of that routeing domain; it should not be used to convey information about the operational status of these systems. The information in the NLRI field is intended to convey static administrative information rather than dynamic transient information: for example, it is not necessary to report that a given system has changed its status from online to offline.

The following guidelines for inclusion of NSAP address prefixes in the NLRI field of UPDATE PDUs originated within a given routeing domain will provide efficient operation of this protocol:

a) NSAP addresses that are within the control of the administrator of a given routeing domain may be reported in the NLRI field for that routeing domain. The NSAP prefixes can provide information about systems that are online, systems that are offline, or unallocated NSAP addresses. The ability to include unallocated NSAP addresses and NSAP addresses of offline systems in the NLRI allows a routeing domain administrator to advertise compact prefixes, thus minimizing the amount of data carried in the BISPDUs.

b) NSAP addresses that are known to correspond to systems that are not under control of the routeing domain administrator should not be included in the NLRI field for that routeing domain. By not listing NSAP address prefixes that identify systems that are not under his control, a routeing domain administrator will comply with the deployment guidelines for ISs and ESs as described in clause 7.2.2, thus assuring correct operation of this protocol.

c) For efficient operation of this protocol, the WITHDRAWN ROUTES field should not be used to report the NLRI of systems in the local routeing domain that are offline. This field should be used only to advertise NSAP prefixes that are no longer under control of the administrator of the local routeing domain, regardless of whether such systems are online or offline.

NOTE 9: Although the protocol in this international standard will operate correctly if each system is reported individually as a maximum-length NSAP address prefix, this will result in a large amount of routeing information, and hence can result in inefficient operation of this protocol.

This protocol provides no means to verify that the preceding guidelines are followed. However, it is within the prerogative of the administrator of any routeing domain to implement policies that ignore UPDATE PDUs that contain an excessive amount of NLRI information or that can cause inefficient operation of this protocol.

7.5 Receive Process

Within the Network layer, the IDRP protocol is located above the ISO 8473 protocol. Therefore, IDRP relies upon ISO 8473 to perform the initial processing of incoming PDUs. After processing the input NPDU, the ISO 8473 protocol machine that resides within the receiving BIS will deliver:

a) BISPDUs to the IDRP Update-Receive Process, or
b) ISO 8473 NPDUs to the IDRP CLNS Forwarding Process.

The ISO 8473 protocol machine shall process inbound NPDUs according to the appropriate ISO 8473 functions.

If the NPDU contains an SPI that identifies IDRP, and the NPDU's source address identifies any system listed in managed objects **INTERNAL-BIS** or **EXTERNAL-BIS-NEIGHBORS**, then the data part of the

NPDU contains a BISPDU. This BISPDU shall be passed to the IDRP finite state machine defined in 7.6.1.

However, if the source address of the NPDU is not an NET listed in the **internalBIS** or the **externalBISNeighbor** attributes of the idrpConfig Managed Object, then:

a) If the NPDU is an OPEN BISPDU without errors, then the BIS shall send a notification ("connectRequestBISUnknown") to Systems Management,
b) If the NPDU is any other BISPDU with or without errors, then the BIS shall send a notification ("PacketBomb") to Systems Management.

The NPDU shall then be discarded.

If the SPI identifies ISO 8473, decapsulate the inner PDU and pass it back to the ISO 8473 protocol machine. (This step permits iterations on multiply encapsulated NPDUs, which may occur, for example, as described in clause 8.4, item b1.)

7.6 BIS-BIS Connection Management

The protocol described in this international standard relies on the underlying Network layer service to establish a full-duplex communications channel between each pair of BISs.

7.6.1 BIS Finite State Machines

A BIS shall maintain one finite state machine (FSM) for each BIS-BIS connection that it supports, and each FSM in a given BIS shall run independently of one another. A BIS-BIS connection will progress through a series of states during its lifetime, which are summarized in the state table shown in Table 2.

BISPDUs passed to this finite state machine are subject the flow control procedures of 7.7.5 if the FSM is in the ESTABLISHED state. When the FSM is in the ESTABLISHED state, only BISPDUs that are not discarded by the flow control process are processed by the FSM. In all other states, all BISPDUs are processed directly by the finite state machine without being subject to flow control procedures.

In describing the FSM transitions in response to receipt of BISPDUs, the following shorthand notation is used:

a) *Receive with no errors* means that the none of the error conditions defined in the appropriate subclause of 7.20 have been detected.
b) *Receive with errors* means that an error condition defined in the appropriate subclause of 7.20 has been detected.

It is possible to receive a BISPDU which is properly formed, but which normally should not be received while the FSM is in the given state. Such an event constitutes an *FSM Error*. If an FSM Error can occur for a given state, it is shown in the description of that state.

7.6.1.1 CLOSED State

Initially the BIS Finite State Machine is in the CLOSED state. The CLOSED state exists when no BIS-BIS connection exists and there is no connection record allocated.

While in the CLOSED state, the BIS shall take the following actions:

a) If the BIS receives a **deactivate**, no action shall be taken and the FSM shall remain in the CLOSED state.
b) If the FSM receives an **activate**, the local BIS shall shall generate an initial sequence number (see 7.7.4), and shall send an OPEN PDU to the remote BIS. The sequence field of the OPEN PDU shall contain the Initial Sequence Number (ISN); the Acknowledgement and Credit Available fields shall contain the value 0; and the Credit Offered field shall contain the initial flow control credit. The FSM shall enter the OPEN-SENT state.
c) If the managed object **ListenForOPEN** is TRUE, and the BIS receives an OPEN PDU with no errors, then the local BIS shall generate an initial sequence number (see 7.7.4) and shall send an OPEN PDU to the remote BIS. The sequence field of the OPEN PDU shall contain the Initial Sequence Number (ISN), the Acknowledgement field shall acknowledge the received OPEN PDU, the Credit Available field shall be set according to the procedures of 7.7.5b, and the Credit Offered field shall contain the initial flow control credit. The FSM shall then change its state to OPEN_RCVD.
d) If the managed object **ListenForOPEN** is TRUE and the BIS receives any BISPDU other than an OPEN PDU with no errors, or if the managed object **ListenForOPEN** is FALSE and the BIS receives any BISPDU, with or without errors, the BIS shall ignore the BISPDU and the FSM shall remain in the CLOSED state.

7.6.1.2 OPEN-SENT State

While in the OPEN-SENT state, the BIS shall take the following actions:

a) If the FSM receives an **activate**, the BIS shall ignore it, and the FSM shall remain in the OPEN-SENT state.

b) If the FSM receives a **deactivate**, the BIS shall send a CEASE PDU to its peer, and the FSM shall enter the CLOSE-WAIT state.

c) If the BIS receives a BISPDU with header errors (see 7.20.1), it shall log the error event, and the FSM shall remain in the OPEN-SENT state.

d) If the BIS receives an OPEN PDU with errors (see 7.20.2), it shall send an IDRP ERROR PDU to the adjacent BIS, acknowledging the remote BIS′s OPEN PDU. The FSM shall then enter the CLOSE-WAIT state.

e) If the BIS receives an OPEN PDU with no errors that does not acknowledge its own previously sent OPEN PDU, then the local BIS shall resend its own OPEN PDU with the same sequence number and with an acknowledgement of the remote BIS′s OPEN PDU. The value of the Credit Available field shall be set according to the procedures of 7.7.5b. The FSM shall then change its state to OPEN-RCVD.

f) If the BIS receives an OPEN PDU with no errors that acknowledges its own previously sent OPEN PDU, the local BIS shall send a KEEPALIVE, RIB REFRESH, or UPDATE PDU that acknowledges the OPEN PDU received from the remote BIS. The FSM shall then enter the ESTABLISHED state.

g) If the BIS receives an IDRP ERROR PDU, either with or without error, it shall send a CEASE PDU, and the FSM shall change its state to CLOSED.

h) If the BIS receives a RIB REFRESH PDU or UPDATE PDU, either with or without errors, it shall issue an IDRP ERROR PDU, indicating ″FSM Error″. The FSM shall then enter the CLOSE-WAIT state.

i) If the BIS receives a KEEPALIVE PDU, it shall issue an IDRP ERROR PDU, indicating ″FSM Error″. The FSM shall then enter the CLOSE-WAIT state.

j) If the BIS receives a CEASE PDU, it shall issue a CEASE PDU in return, and then the FSM shall enter the CLOSED state.

k) If the BIS does not receive an OPEN PDU within a period t_R after sending an OPEN PDU, the BIS shall resend the OPEN PDU. If the OPEN PDU is retransmitted n times, the local BIS shall issue a **deactivate** to close the BIS-BIS connection.

NOTE 10: The value t_R should be chosen to be large enough so that attempting to establish a connection to an unresponsive peer BIS does not consume significant network resources. The values of t_R and n must be chosen so that $t_R \times n$ is greater than the architectural constant **CloseWaitDelay**.

Table 2 (Page 1 of 2). BIS Finite State Machine. This table summarizes the effects that its inputs will have on an IDRP FSM, giving both state transitions and the actions to be taken.

STATE → / INPUT ↓	CLOSED	OPEN-RCVD	OPEN-SENT	CLOSE-WAIT	ESTABLISHED
activate	S=OPEN-SENT A=send OPEN PDU	S=OPEN-RCVD A=none	S=OPEN-SENT A=none	S=CLOSE-WAIT A=none	S=ESTABLISHED A=none
deactivate	S=CLOSED A=none	S=CLOSE-WAIT A=send CEASE PDU	S=CLOSE-WAIT A=send CEASE PDU	S=CLOSE-WAIT A=none	S=CLOSE-WAIT A=send CEASE PDU
Expiration of CloseWaitDelay Timer	S=CLOSED A=none	S=OPEN-RCVD A=none	S=OPEN-SENT A=none	S=CLOSED A=none	S=ESTABLISHED A=none
Expiration of Hold Timer	S=CLOSED A=none	S=OPEN-RCVD A=none	S=OPEN-SENT A=none	S=CLOSE-WAIT A=none	S=CLOSE-WAIT A=Send IDRP ERROR PDU to report error
Receive BISPDU with Header Error	S=CLOSED A=none	S=OPEN-RCVD A=log error event	S=OPEN-SENT A=log error event	S=CLOSE-WAIT A=none	S=ESTABLISHED A=log error event
Receive KEEPALIVE PDU with no errors	S=CLOSED A=none	If ACK is correct, S=ESTABLISHED A=Restart Hold Timer If ACK is incorrect, S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report FSM Error	S=CLOSE-WAIT A=Send IDRP ERROR PDU to report FSM Error	S=CLOSE-WAIT A=send CEASE, restart CloseWaitDelay timer	S=ESTABLISHED A=Restart Hold Timer
Receive CEASE PDU with no errors	S=CLOSED A=none	S=CLOSED A=send CEASE	S=CLOSED A=send CEASE	S=CLOSED A=none	S=CLOSED A=send CEASE
Receive OPEN PDU with errors	S=CLOSE A=none	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report OPEN PDU error	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report OPEN PDU error	S=CLOSE-WAIT A=none	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report OPEN PDU error
Receive OPEN PDU with no errors	If ListenForOPEN is TRUE, S=OPEN-RCVD A=send OPEN PDU If ListenForOPEN is FALSE, S=CLOSED A=none	If ACK is correct, S=ESTABLISHED A=send KEEPALIVE, UPDATE, or RIB REFRESH PDU If ACK is incorrect, S=OPEN-RCVD, A=send OPEN PDU	If ACK is correct, S=ESTABLISHED A=send KEEPALIVE, UPDATE, or RIB REFRESH PDU If ACK is incorrect, S=OPEN-RCVD A=send OPEN PDU	S=CLOSE-WAIT A=send CEASE, restart CloseWaitDelay timer	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report FSM error
Receive UPDATE PDU with errors	S=CLOSED A=none	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report UPDATE PDU error	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report FSM error	S=CLOSE-WAIT A=none	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report UPDATE PDU error

Table 2 (Page 2 of 2). BIS Finite State Machine. This table summarizes the effects that its inputs will have on an IDRP FSM, giving both state transitions and the actions to be taken.

STATE → / INPUT ↓	CLOSED	OPEN-RCVD	OPEN-SENT	CLOSE-WAIT	ESTABLISHED
Receive UPDATE PDU with no errors	S=CLOSED A=none	If ACK is correct, S=ESTABLISHED A=7.14, restart Hold Timer If ACK is incorrect, S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report FSM Error	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report FSM error	S=CLOSE-WAIT A=send CEASE, restart CloseWaitDelay timer	S=ESTABLISHED A=7.14, restart Hold Timer
Receive IDRP ERROR PDU with errors	S=CLOSED A=none	S=CLOSED A=Send CEASE PDU	S=CLOSED A=Send CEASE PDU	S=CLOSED A=Send CEASE PDU	S=CLOSE-WAIT A=Send CEASE PDU
Receive IDRP ERROR PDU with no errors	S=CLOSED A=none	S=CLOSED A=Send CEASE PDU	S=CLOSED A=Send CEASE PDU	S=CLOSED A=send CEASE	S=CLOSED A=Send CEASE PDU
Receive RIB REFRESH PDU with errors	S=CLOSED A=none	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report RIB REFRESH PDU error	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report FSM error	S=CLOSE-WAIT A=none	S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report RIB REFRESH PDU error
Receive RIB REFRESH PDU with no errors	S=CLOSED A=none	If ACK is correct, S=ESTABLISHED A=7.10.3, restart Hold Timer If ACK is incorrect, S=CLOSE-WAIT A=Send IDRP ERROR PDU to peer BIS to report FSM Error	S=CLOSE-WAIT A=Send IDRP ERROR PDU to report FSM Error	S=CLOSE-WAIT A=send CEASE, restart CloseWaitDelay timer	S=ESTABLISHED A=7.10.3, restart Hold Timer

NOTES:

a) ″S″ indicates the state into which the FSM will make a transition after performing the indicated action.

b) ″A″ indicates the action to be taken.

c) ″X.Y.Z″ is shorthand notation for ″do as specified in clause X.Y.Z″.

d) The phrase ″no errors″ for a given BISPDU type means that no condition described in the appropriate subclause of 7.20 has been detected.

e) The phrase ″with errors″ for a given BISPDU type means that a condition described in the appropriate subclause of 7.20 has been detected.

f) Since the KEEPALIVE PDU and the CEASE PDU consist of only a fixed BISPDU header, errors in these BISPDUs are handled as Header Errors. Hence, there are no explicit entries in the table for ″KEEPALIVE with errors″ or ″CEASE with errors″.

7.6.1.3 OPEN-RCVD State

While in the OPEN-RCVD state, the BIS shall take the following actions:

a) If the BIS receives an **activate**, it shall ignore it and the FSM shall remain in the OPEN-RCVD state.

b) If the BIS receives a **deactivate**, it shall send a CEASE PDU to the remote BIS, and the FSM shall enter the CLOSE-WAIT state.

c) If the BIS receives a BISPDU with a header error, it shall log the error event, and the FSM shall remain in the OPEN-RCVD state.

d) If the BIS receives a KEEPALIVE PDU that acknowledges its previously sent OPEN PDU, then the FSM shall enter the ESTABLISHED state.

e) If the BIS receives a KEEPALIVE PDU that does not acknowledge its previously sent OPEN PDU, the BIS shall issue an IDRP ERROR PDU indi-

cating "FSM Error", and the FSM shall change its state to CLOSE-WAIT.

f) If the BIS receives a CEASE PDU, it shall issue a CEASE PDU in return, and then the FSM shall enter the CLOSED state.

g) If the BIS receives an OPEN PDU with no errors from the remote BIS that acknowledges the local BIS's previously sent OPEN PDU, the BIS shall send a KEEPALIVE, RIB REFRESH, or UPDATE PDU that acknowledges the OPEN PDU received from the remote BIS. The FSM shall then enter the ESTABLISHED state.

h) If the BIS receives an OPEN PDU with no errors that does not acknowledge the local BIS's previously sent OPEN PDU, then the local BIS shall resend its own OPEN PDU with the same sequence number, and shall also include an acknowledgement of the remote BIS's OPEN PDU. The FSM shall remain in the OPEN-RCVD state.

i) If the BIS receives an UPDATE PDU or RIB REFRESH PDU with errors, the shall send an IDRP ERROR PDU to the remote BIS, and the FSM shall enter the CLOSE-WAIT state.

j) If the BIS receives an UPDATE PDU or RIB REFRESH PDU with no errors that acknowledges the OPEN PDU previously sent by the local BIS, the FSM shall enter the ESTABLISHED state.

k) If the BIS receives an UPDATE PDU or RIB REFRESH PDU with no errors that does not acknowledge the OPEN PDU previously sent by the local BIS, the BIS shall issue an IDRP ERROR PDU indicating "FSM Error", and the FSM shall change its state to CLOSE-WAIT.

l) If the BIS receives an IDRP ERROR PDU, either with or without errors, it shall send a CEASE PDU to the remote BIS, and the FSM shall enter the CLOSED state.

m) If the BIS does not exit the OPEN-RCVD state within a period t_R after sending an OPEN PDU, the BIS shall resend the OPEN PDU. If the OPEN PDU is transmitted n times, the local BIS shall issue a **deactivate**.

NOTE 11: The value t_R should be chosen to be large enough so that attempting to establish a connection to an unresponsive peer BIS does not consume significant network resources. The values of t_R and n must be chosen so that $t_R \times n$ is greater than the architectural constant **CloseWaitDelay**.

7.6.1.4 ESTABLISHED State

The ESTABLISHED state is entered from either the OPEN-SENT or the OPEN-RCVD states. It is entered when a connection has been established by the successful exchange of state information between two sides of the connection. Each side has exchanged and received such data as initial sequence number, maximum PDU size, credit offered, protocol version number, hold time, and RDI of the other side. In addition, the remote BIS may also have been authenticated.

In ESTABLISHED state, both BISs that are involved in the connection may exchange UPDATE PDUs, KEEPALIVE PDUs, IDRP ERROR PDUs, RIB REFRESH PDUs, and CEASE PDUs.

While in the ESTABLISHED state, the local BIS shall take the following actions:

a) If the FSM receives an **activate**, the FSM shall ignore it, and the FSM shall remain in the ESTABLISHED state.

b) If the FSM receives a **deactivate**, the BIS shall send a CEASE PDU to the peer BIS. The FSM shall enter the CLOSE-WAIT state.

c) If the Hold Timer expires, the BIS shall issue an IDRP ERROR PDU to the remote BIS, reporting a Hold_Timer error. The FSM shall enter the CLOSE-WAIT state.

d) If the BIS receives a BISPDU with a header error, it shall log the error event, and the FSM shall remain in the ESTABLISHED state.

e) If the BIS receives a KEEPALIVE PDU, it shall restart its Hold Timer, and the FSM shall remain in the ESTABLISHED state.

f) If the BIS receives a CEASE PDU, it shall issue a CEASE PDU in return, and then the FSM shall enter the CLOSED state.

g) If an OPEN PDU with no errors is received from the peer BIS, it shall issue an IDRP ERROR PDU, indicating FSM error. The FSM shall enter the CLOSE-WAIT state.

h) If the BIS receives an UPDATE PDU with no errors, the BIS shall perform the actions provided in clause 7.14, and shall restart its Hold Timer. The FSM shall remain in the ESTABLISHED state.

i) If the BIS receives a RIB REFRESH PDU with no errors, the BIS shall perform the actions provided in clause 7.10.3, and shall restart its Hold Timer. The FSM shall remain in the ESTABLISHED state.

j) If the BIS receives an UPDATE PDU with errors, an OPEN PDU with errors, or a RIB REFRESH PDU with errors, it shall send an IDRP ERROR PDU to

the remote BIS to report the error, and the FSM shall enter the CLOSE-WAIT state.

k) If the BIS receives an IDRP ERROR PDU, either with or without errors, it shall send a CEASE PDU to the remote BIS. The FSM shall enter the CLOSED state.

7.6.1.5 CLOSE-WAIT State

When an FSM enters the CLOSE-WAIT state, the local BIS is preparing to close the connection with the remote BIS. Upon entering this state, the local BIS shall mark all entries in the Adj-RIB-In associated with the adjacent BIS as unreachable, and shall then re-run its Decision Process. The CloseWaitDelay timer shall be started.

While in the CLOSE-WAIT state, the BIS shall take the following actions:

a) If the CloseWaitDelay timer expires, the connection ceases to exist. The FSM shall enter the CLOSED state.

b) If the BIS receives a CEASE PDU, the FSM shall enter the CLOSED state.

c) If the BIS receives an IDRP ERROR PDU, it shall send a CEASE PDU to the peer BIS. The FSM shall then enter the CLOSED state.

d) If the BIS receives any other type of BISPDU, with or without errors, it shall issue a CEASE PDU. The FSM shall remain in the CLOSE-WAIT state, and the CloseWaitDelay timer shall be restarted.

e) The BIS shall take no action for any of the following inputs, and the FSM shall remain in the CLOSE-WAIT state:

— activate
— deactivate
— Expiration of Hold Timer

7.6.2 Closing a Connection

The closing of a connection can be initiated by a **deactivate** generated by the local system, by receipt of an incorrect PDU, by receipt of a IDRP ERROR PDU, by expiration of the Hold Timer, or by receipt of a CEASE PDU. The actions taken in response to each of these stimuli are shown in Table 2

When the connection enters the CLOSED state, all routes that had been exchanged between the pair of BISs are implicitly withdrawn from service; hence, the local BIS should rerun its Decision Process.

7.7 Validation of BISPDUs

The protocol described in this international standard is a connection oriented protocol in which the underlying Network Layer service is used to establish full-duplex communication channels between pairs of BISs, as described in clause 7.6. This international standard supports the use of any of the following three mechanisms for validating BISPDUs. Types 1,2, and 3 provide data integrity for the contents of BISPDUs; in addition, types 2 and 3 provide peer BIS authentication. Each mechanism is described below. Figure 6 illustrates the data integrity mechanisms used for authentication types 1 and 3; Figure 10 illustrates an example approach that might be used to provide both data integrity and peer BIS authentication for authentication type 2.

7.7.1 Authentication Type 1

For all BISPDUs that flow on a connection that was established in response to an OPEN PDU whose authentication code field was equal to 1, the validation field shall contain a 16-octet unencrypted checksum:

a) Generating a Validation Pattern:

The contents of the Validation Pattern field that is included in an outbound BISPDU shall be generated by applying the algorithm of Annex C to the input data stream that consists of the contents of the entire BISPDU with all bits of the Validation Pattern field initially set to 0. The output of this step is an unencrypted 16-octet long checksum, which shall be placed in the Validation Pattern field of the BISPDU.

b) Checking the Validation Pattern of an Inbound BISPDU:

The contents of the Validation Pattern field of an inbound BISPDU shall be checked by applying the algorithm of Annex C to the contents of the inbound BISPDU with its Validation Pattern set to all zeros. Call this quantity the ″reference pattern″.

If the ″reference pattern″ matches the contents of the Validation Pattern field of the inbound BISPDU, then the BISPDU's checksum is correct; otherwise, it is incorrect.

7.7.2 Authentication Type 2

When the authentication type code of the OPEN PDU is 2, the pattern carried in the 16-octet Validation Pattern field of the fixed header shall provide both peer-BIS authentication and data integrity for the contents of the BISPDU. The specific mechanisms used to provide these functions are not specified by this international standard. However, they must be

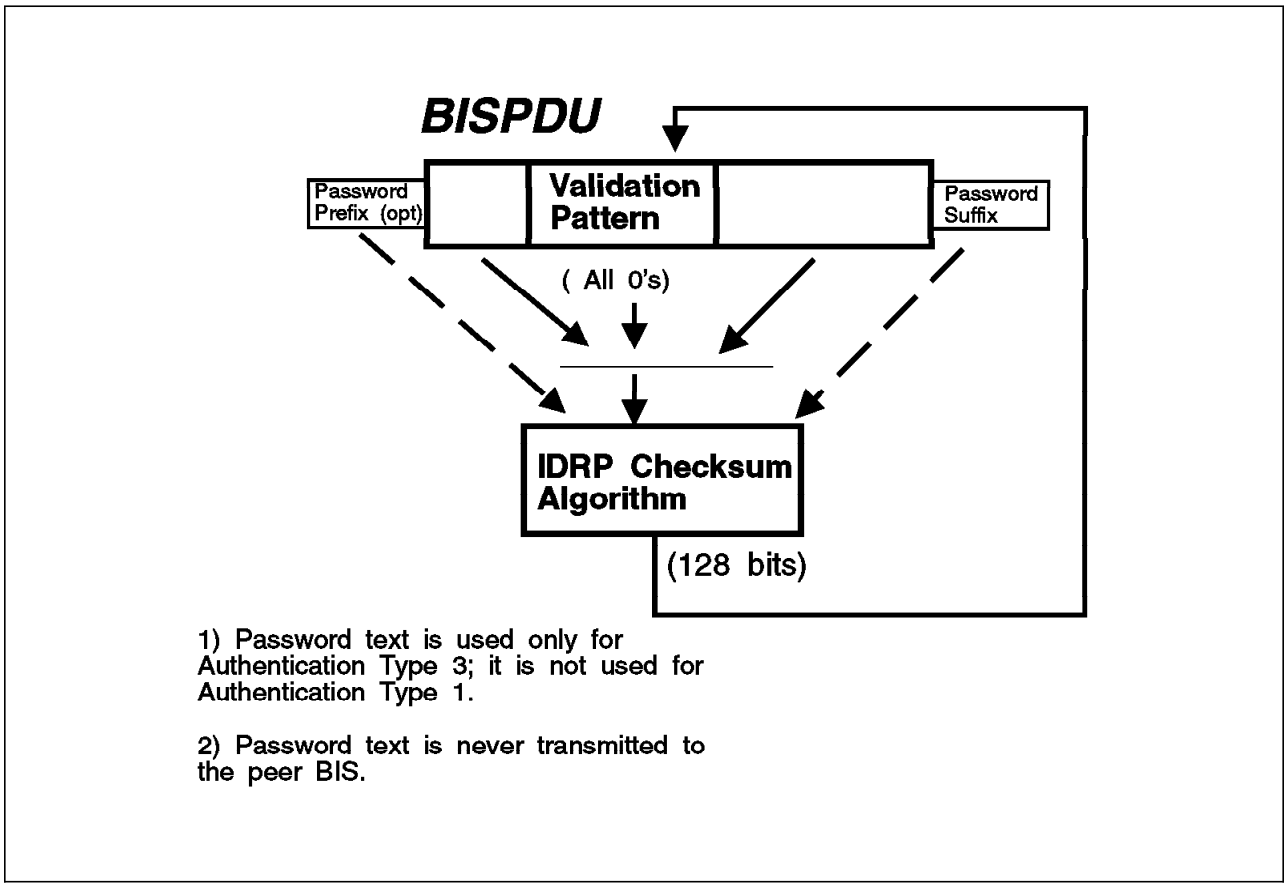

Figure 6. Illustration of Authentication Types 1 and 3

agreed to by the pair of communicating BISs as part of their security association.

An example of type 2 authentication that uses an encrypted version of MD4 checksum is described in Annex D.

NOTE 12: This international standard includes as an optional function a mechanism that can be used for authentication of the source of a BISPDU. Other security-related facilities (for example, protection against replay of BISPDUs or the ability to re-key during a BIS_BIS connection) are not intended to be provided by this protocol, and therefore are not specified in this International Standard.

All of the relevant security services identified in ISO 7498-2, including authentication, could be achieved by the use of CD 11577, a security protocol specified for the provision of secure ISO 8473 communications (for example, see 9.1).

7.7.3 Authentication Type 3

When the authentication type code of the OPEN PDU is 3, the Validation Pattern field shall contain a 16-octet checksum covering both the contents of the BISPDU and some additional Password Text, which is not transmitted to the peer BIS. The checksum provides data integrity and the untransmitted Password Text provides peer BIS authentication.

The mechanisms are as follows:

a) Generating a Validation Pattern:

The contents of the Validation Pattern field that is carried in the outbound BISPDU shall be generated by the following process, which is illustrated in Figure 6.

1) Password text shall be appended to the BISPDU immediately after the final octet of the BISPDU (as defined by the BISPDU length field of the BISPDU header). Additional password text may also be prepended to the BISPDU immediately prior to the first octet of its header.
2) A checksum that covers the concatenated contents of the BISPDU and the password text shall be generated using the methods of Annex C with all bits of the Validation Pattern initially set to zero. The resultant checksum shall then be placed in the Validation Pattern field of the BISPDU.
3) The password text shall not be transmitted along with the BISPDU.

b) Checking the Validation Pattern of an Inbound BISPDU:

The contents of the Validation Pattern field of an inbound BISPDU shall be checked by the following procedure:

1) Append the Password Text to the BISPDU immediately after the final octet of the BISPDU (as defined by the BISPDU Length field of the BISPDU header. If additional Password text was also prepended to the BISPDU by the sender, then prepend this text immediately prior to the first octet of the BISPDU.
2) Apply the IDRP Checksum Algorithm to the data stream that consists of the concatenated contents of the BISPDU and the password text, with all bits of the BISPDU Validation Pattern set to zero. Call this value the ″reference pattern″.
3) If the ″reference pattern″ is identical to the data carried in the Validation Pattern of the incoming BISPDU, then the peer BIS has been authenticated. If the ″reference pattern″ does not match the Validation Pattern, the receiving BIS shall inform system management that an authentication failure has occurred. The incoming BISPDU shall be ignored. The receiving BIS shall not send an IDRP ERROR PDU to the peer BIS because the identity of the peer has not been authenticated.

7.7.4 Sequence Numbers

A sequence number is a 4-octet unsigned value. Sequence numbers shall increase linearly from 1 up to a maximum value of $2^{32} - 1$. The value 0 is not a valid sequence number.

The rules for manipulating sequence numbers are:

a) When a BIS initially establishes a connection with an adjacent BIS, the first sequence number shall be set to 1 and shall increase linearly to a value of $2^{32} - 1$. Before attempting to establish an initial BIS-BIS connection with an adjacent BIS, the local BIS must ensure that it has not sent a BISPDU to the adjacent BIS for at least **CloseWaitDelay** seconds.

b) The sequence number shall not be incremented for the KEEPALIVE PDU, CEASE PDU, and the IDRP ERROR PDU.

c) If the connection is subsequently closed under the conditions described in Table 2 and a subsequent connection is to be made to the same adjacent BIS, the local BIS shall, as a local matter, choose one of the following options:

— Maintain status of the sequence number space, and use any value greater than the last-used value, or
— Ensure that at least **CloseWaitDelay** seconds have passed since the last BISPDU was sent to the adjacent BIS, and start with any sequence number. The choice of the initial value of the sequence number is a local matter.

d) After a BIS sends a BISPDU with the maximum permissible sequence number ($2^{32} - 1$) the BIS shall not send any further BISPDUs until the BISPDU with maximum sequence number and all outstanding BISPDUs have been acknowledged using the procedure of 7.7.5. The BIS then shall set its *lower window edge* (see 7.7.5) to one. When a BIS receives a BISPDU with a sequence number of one, after having acknowledged a BISPDU with the maximum permissible sequence number, it shall set the value of its *next expected sequence number* to one, prior to processing that BISPDU.

NOTE 13: The maximum lifetime of an 8473 NPDU is 128 seconds. Since the architectural constant **CloseWaitDelay** is 150 seconds, it can be guaranteed that all outstanding BISPDUs (which are carried as user data within an encapsulating 8473 NPDU) will have expired before the new BIS-BIS connection is established.

7.7.5 Flow Control

After an IDRP connection is established, the BIS Finite State Machine is in state ESTABLISHED (see section 7.6.1), and flow control and packet sequencing is in effect. The IDRP flow control process shall obey the following rules:

a) A separate series of sequence numbers shall be maintained for each direction of a BIS-BIS connection, with the initial sequence number value chosen by the sender of a BISPDU and declared in the *Sequence* field of its OPEN PDU. The local BIS will maintain a window to manage transmission of BISPDUs to the remote BIS. The sender′s *lower window edge* shall be set to the initial sequence number plus one; the sender′s *upper window edge* shall be set to the *lower window edge* plus the value of *credit offered* contained in the peer BIS′s OPEN PDU. Record is also kept of the *next expected sequence number* for an inbound UPDATE, RIB REFRESH, KEEPALIVE, or OPEN PDU to be received from the peer BIS; this is initially set to the value of one plus *Sequence* that is carried in the peer BIS′s OPEN PDU.

b) An UPDATE PDU or RIB REFRESH PDU shall not be sent if the upper window edge is less than or equal to the lower window edge. When a BISPDU is sent, the value of *Sequence* in the fixed header

shall be set to the current value of the lower window edge. When an UPDATE or RIB REFRESH PDU is to be sent, the local BIS shall generate the contents of the BISPDU based on the current value of the lower window edge. The local BIS shall increment the local window edge by one before it transmits the BISPDU to the peer BIS and before it generates any other BISPDUs or processes any received BISPDUs; when a BISPDU other than an UPDATE or RIB REFRESH PDU is to be sent, the lower window edge shall not be incremented. The value of *Acknowledgement* shall be set to the value of the next expected sequence number less one. The value of *credit offered* shall be set to the number of additional BISPDUs that the local BIS is currently able to accept from the peer BIS. Credit, once offered, can not be revoked (that is, the remote BIS′s upper window edge can not be reduced). Therefore, the sum of *Acknowledgement* and *credit offered* must never decrease in successive BISPDUs. The value of *credit available* shall be set to the upper window edge less the lower window edge (after incrementing the lower window edge, if appropriate).

The local BIS shall retain a copy of transmitted UPDATE and RIB REFRESH BISPDUs for possible retransmission.

c) An incoming UPDATE PDU or RIB REFRESH PDU whose *Sequence* value corresponds to the next expected sequence number shall be accepted and passed to the Finite State Machine described in 7.6.1; the next expected sequence number shall be incremented by one.

An incoming UPDATE PDU or RIB REFRESH PDU whose *Sequence* is less than the next expected sequence number shall be discarded. An incoming UPDATE PDU or RIB REFRESH PDU whose *Sequence* is greater than the next expected sequence number shall be discarded, unless re-ordering is supported as a local implementation option, and the sequence number is not greater than the peer′s *upper window edge*.

An incoming KEEPALIVE PDU or OPEN PDU whose *Sequence* value corresponds to the next expected sequence number shall be accepted and passed to the Finite State Machine described in 7.6.1. An incoming KEEPALIVE PDU or OPEN PDU whose *Sequence* does not correspond to the next expected sequence number shall be discarded.

An Incoming CEASE PDU or IDRP ERROR PDU shall be accepted and passed to the Finite State Machine described in 7.6.1 regardless of its *Sequence* value.

Whenever a BIS receives an UPDATE PDU, RIB REFRESH PDU, or KEEPALIVE PDU, it shall inspect its *Acknowledgement* and *credit offered* fields. Any BISPDUs retained for retransmission whose sequence number is less than or equal to the value of the *Acknowledgement* field shall be discarded. If the sum of one plus the value of *Acknowledgement* plus the value of *credit offered* in the received BISPDU is greater than the local BIS′s current upper window edge, then the BIS shall set its upper window edge to this sum.

d) A BIS shall acknowledge receipt of incoming UPDATE PDUs and RIB REFRESH PDUs within a period t_A of their receipt. The acknowledgement may be accomplished by means of an UPDATE PDU or a RIB REFRESH PDU sent as outlined in item b above. However, if no UPDATE PDU or RIB REFRESH PDU is available to be sent, then a KEEPALIVE PDU may be sent instead, with its *Sequence* set to the lower window edge and its *Acknowledgement*, *credit offered*, and *credit available* set as in step b above.

e) If a retained BISPDU remains unacknowledged after a period t_R, then it shall again be transmitted and again retained for possible retransmission. If, for a retained BISPDU, t_R expires after n retransmissions, the local BIS shall issue a **deactivate** to close the BIS-BIS connection.

NOTE 14: The value t_R should be chosen to be greater than the value $t_A + 2 \times L$, where L is the transmission delay over the subnetwork or virtual link between the pair of communicating BISs.

f) The local BIS shall provide its peer BIS with sufficient credit to send further BISPDUs as long as the local BIS has resources to receive them. Therefore, if the local BIS receives a BISPDU whose *credit available* is equal to zero (that is, the peer BIS believes itself unable to send additional BISPDUs), then as soon as resources are available locally, the local BIS shall send an UPDATE PDU or a RIB REFRESH PDU, if appropriate. If not, then a KEEPALIVE PDU shall be sent.

NOTE 15: An UPDATE PDU of minimal size will contain the Unfeasible Route Count field with a value of zero, but will not contain any path attributes or NLRI. Thus, its size will be only 33 octets.

A KEEPALIVE PDU that advertises a non-zero value of credit offered in response to a received BISPDU with a credit available of zero shall be retransmitted within a period t_R until the local BIS receives any in-sequence BISPDU that reports a non-zero value of credit available. If t_R expires after n retransmissions, then the local BIS shall issue a **deactivate** to close the connection.

g) A BIS that sends a BISPDU with zero credit available shall respond within a period t_A to a BISPDU that causes the upper window edge to be

increased. The response shall consist of an UPDATE PDU or a RIB REFRESH PDU, if available, or a KEEPALIVE PDU, if not.

h) A BIS that has not sent any BISPDU for a period t_I shall send a KEEPALIVE PDU, with *Sequence* equal to the lower window edge, and *Acknowledgement*, *credit offered*, and *credit available* set as in step b above.

NOTE 16: The condition $t_I >> t_R$ should be satisfied, where t_I is one third of the Hold Timer value.

i) A BIS that has sent a BISPDU containing a *credit offered* of zero shall, as soon as its local resources become available to process additional BISPDUs from its peer, send an UPDATE PDU or RIB REFRESH PDU, if appropriate, containing a non-zero value of *credit offered*. If neither of these BISPDU types is appropriate, then a KEEPALIVE PDU shall be sent.

j) The BIS shall issue a **deactivate** to close the BIS-BIS connection if no BISPDUs are received for a period equal to the value of Hold Time that is carried in the OPEN PDU.

7.8 Version Negotiation

BIS peers may negotiate the version number of IDRP by making successive attempts to open a BIS-BIS connection, starting with the highest supported version number (contained in managed object **version**) and decrementing the number each time a connection attempt fails. The lack of support for a particular IDRP version is indicated by an IDRP ERROR PDU with error code ″OPEN_PDU_Error″ and an error subcode of ″Unsupported_Version_Number″. One BIS may determine the highest version number supported by the other BIS (as advertised in its OPEN PDU) by examining the ″Data″ field of the received IDRP ERROR PDU. No further retries should be attempted if the version number reaches zero.

7.9 Checksum Algorithm

The checksums used in this international standard for authentication types 1 and 3 shall be generated in accordance with the procedures described in normative Annex C. For an input data stream of any length, this algorithm will generate a checksum that is 16 octets long. This algorithm shall be used to generate the checksums for both the BISPDUs and the RIBs.

7.10 Routeing Information Bases

The Routeing Information Base (RIB) within a BIS consists of three distinct parts, as shown in Figure 7:

a) **Adj-RIBs-In:** The Adj-RIBs-In store routeing information that has been learned from inbound UPDATE PDUs. Their contents represent routes that are available as input to the Decision Process. A BIS must support at least one Adj-RIB-In for each of its neighbor BISs; it may optionally support several Adj-RIBs-In for a given neighbor BIS. Within the set of Adj-RIBs-In associated with a given neighbor BIS, no two shall have the same RIB-Att (see clause 7.10.1).

b) **Loc-RIBs:** The Loc-RIBs contain the local routeing information that the BIS has selected by applying its local policies to the routeing information contained in its Adj-RIBs-In. A BIS may support multiple Loc-RIBs. No two Loc-RIBs within a given BIS shall have the same RIB-Att (see clause 7.10.1). Information in the Loc-RIB is used to build the Adj-RIBs-Out.

c) **Adj-RIBs-Out**: The Adj-RIBs-Out store the information that the local BIS has selected for advertisement to its neighbors. A BIS must support at least one Adj-RIB-Out for each of its neighbor BISs; it may optionally support several Adj-RIBs-Out for a given neighbor BIS. Within the set of Adj-RIBs-Out associated with a given neighbor BIS, no two shall have the same RIB-Att (see clause 7.10.1). The routeing information stored in the Adj-RIBs-Out will be carried in the local BIS′s UPDATE PDUs and advertised to its neighbor BISs.

In summary, the Adj-RIBs-In contain unprocessed routeing information that has been advertised to the local BIS by its neighbors; the Loc-RIBs contain the routes that have been selected by the local BIS′s Decision Process; and the Adj-RIBs-Out organize the selected routes for advertisement to specific neighbor BISs by means of the local BIS′s UPDATE PDUs.

NOTE 17: Although the conceptual model distinguishes between Adj-RIBs-In, Adj-RIBs-Out, and Loc-RIBs, this does neither implies nor requires that an implementation must maintain three separate copies of the routeing information. The choice of implementation (for example, 3 copies of the information vs. 1 copy with pointers) is not constrained by this standard.

7.10.1 Identifying an Information Base

Each information base (a single Adj-RIB-In, a single Loc-RIB, or a single Adj-RIB-Out) has one and only one RIB-Att associated with it. A RIB-Att is composed of a set of Distinguishing Attributes that the local BIS supports: in particular, a RIB-Att may consist of one or more Distinguishing Attributes that form a permissible combination, as defined in clause 7.11.2.

The managed object **RIB-AttsSet** explicitly enumerates all the RIB-Atts that a BIS supports. Managed

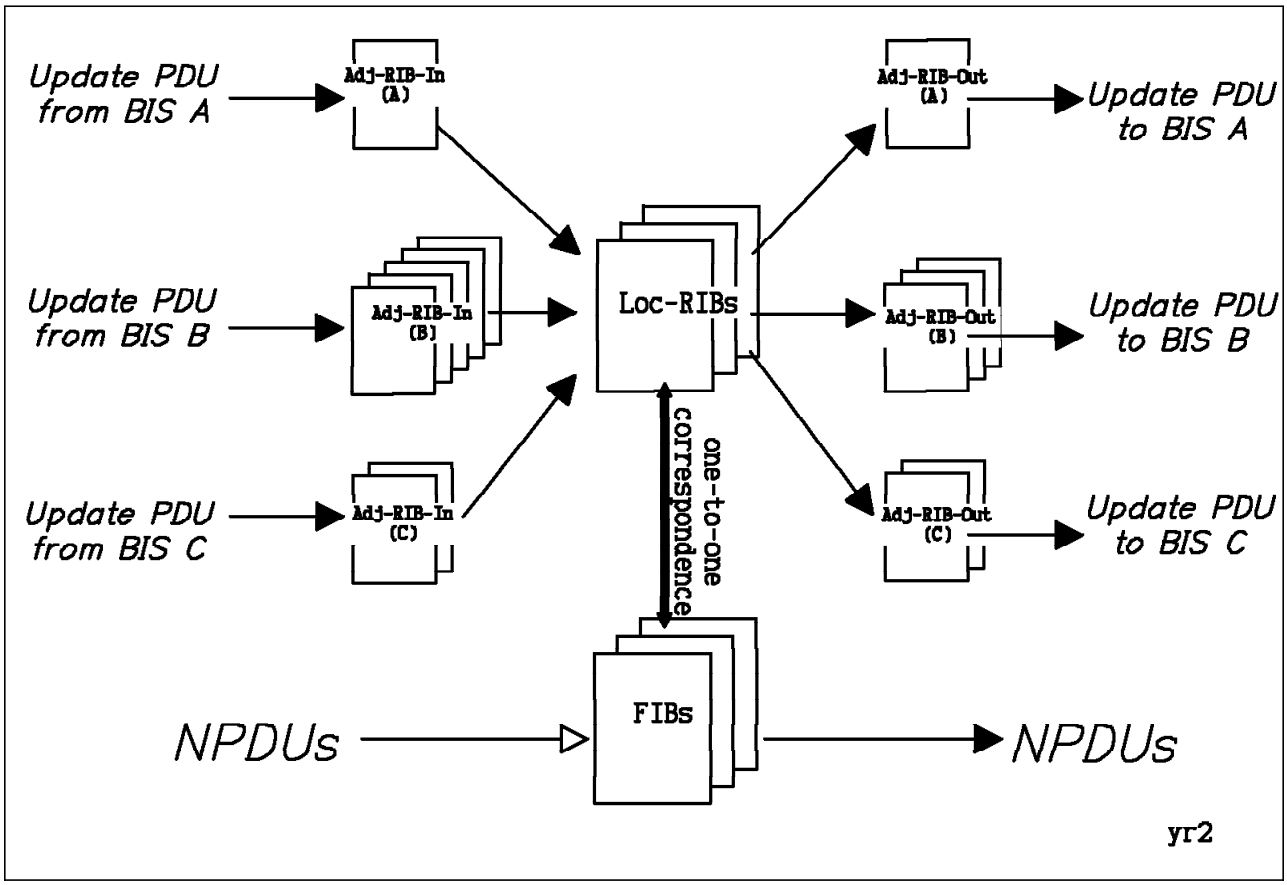

Figure 7. Routeing Information Base. An RIB is comprised of Adj-RIBs-In, Adj-RIBs-Out, and Loc-RIBs

object **RIB-AttsSet** shall not contain any pairs of RIB-Atts that are identical, thus assuring that each RIB-Att is unambiguous within the BIS.

All BISs located within a given routeing domain shall support the same RIB-Atts: that is, the managed object **RIB-AttsSet** of every BIS within an RD shall list the same RIB-Atts. When a BIS receives an OPEN PDU from another BIS located in its own routeing domain, it shall compare the information in the field *RIB-AttsSet* with the information in its local managed object **RIB-AttsSet**. If they do not match, then the appropriate error handling procedure in clause 7.20.2 shall be followed.

Each BIS shall support default information bases (Adj-RIBs-In, Adj-RIBs-Out, Loc-RIB, and FIB) that correspond to the RIB-Att that is composed of an empty set of Distinguishing Attributes.

NOTE 18: Because policy is a local matter, IDRP does not specify the criterion used to select the information to be placed in the default Loc-RIB. However, since the following mandatory path attributes are present in every route, it is suggested that RD_PATH and RD_HOP_COUNT should be used for this purpose.

7.10.2 Validation of RIBs

A BIS shall not continue to operate for an extended period with corrupted routeing information. Therefore, the BIS shall operate in a ″fail-stop″ manner: when corruption of a RIB is detected, the BIS shall immediately take action to cease using the routes contained in the corrupted information base.

In the absence of an implementation-specific method for insuring this, the BIS shall perform the following checks at least every **maxRIBIntegrityCheck** seconds:

a) Upon expiration of its **maxRIBIntegrityCheck** timer, the BIS shall recheck the checksum of the routeing information contained in each of its Adj-RIBs-In in order to detect corruption of routeing information while in memory.

   If corruption is detected, the BIS shall purge the Adj-RIB-In, and shall notify System Management of a ″Corrupted AdjRIBIn″ event.

   NOTE 19: This standard does not prescribe a specific checksum algorithm but notes that the procedures described in Annex F satisfy the requirements given above. Other approaches can also be used: for example, it may be possible to use an incremental algo-

rithm to compute the checksum for a given Adj-RIB-In when new information is received.

After detection of a corrupted Adj-RIB-In, a BIS may choose to issue a RIB REFRESH PDU, asking for a solicited refresh of the routeing information from its peer BIS.

b) On completion of its check of the Adj-RIBs-In, the BIS shall rerun its Decision Process, regardless of whether or not corruption of the Adj-RIBs-In has been detected. As a byproduct of running the Decision Process, the BIS will construct new information for its Loc-RIBs, and will then regenerate its Adj-RIBs-Out and its FIBs. Thus, any corrupted information that may have been present in the Adj-RIBs-Out or the FIBs will be replaced as a result.

c) Upon completion of these checks, the BIS shall reset the timer to the value **MaxRIBIntegrityCheck** with jitter applied in accordance with clause 7.17.3.3.

Since a given Adj-RIB-In that had been corrupted will have been purged before the Decision Process is re-executed, the defective information will not be used in the recalculation.

An explicit integrity check on the contents of the Loc-RIBs, Adj-RIBs-Out, and the FIBs is not required, since corrupted information will be replaced periodically when the Decision Process is re-run.

As a local option, a BIS may also choose to perform an explicit integrity check on the routeing information in its Loc-RIBs, Adj-RIBs-Out, and FIBs. If such an integrity check detects that the information base has become corrupted, then the BIS shall immediately rerun its Decision Process, and should notify System Management of ″Corrupt Loc-RIB″, ″CorruptAdjRIBOut″, or ″CorruptFIB″, as appropriate.

7.10.3 Use of the RIB REFRESH PDU

The RIB REFRESH PDU can be used by a BIS to solicit a refresh of its Adj-RIBs-In by a neighbor BIS, or to send an unsolicited refresh to a neighbor BIS:

a) *Solicited Refresh*

A BIS may request a neighbor BIS to refresh one or more of the local BIS′s Adj-RIBs-In by sending a RIB-REFRESH PDU that contains the OpCode for RIB-Refresh-Request and the RIB-Atts of the Adj-RIBs-In that it wants to be refreshed.

When the neighbor BIS receives a RIB-REFRESH PDU with OpCode RIB-Refresh-Request, it shall send back a RIB-REFRESH PDU with OpCode RIB-Refresh-Start, followed by a sequence of UPDATE PDUs that contain the information in its Adj-RIBs-Out associated with the requesting BIS. The neighbor BIS shall indicate the completion of the refresh by sending a RIB-REFRESH PDU with OpCode RIB-Refresh-End.

b) *Unsolicited Refresh*

A BIS may initiate an unsolicited refresh by sending a RIB-REFRESH PDU with OpCode RIB-Request-Start, followed by a sequence of UPDATE PDUs that contain the information in its Adj-RIBs-Out that been advertised to a given BIS. The completion of the refresh shall be indicated by sending the RIB-REFRESH PDU with OpCode RIB-Refresh-End.

When a BIS receives a RIB REFRESH PDU with OpCode 2 (RIB Refresh Start), it shall not change any of the routeing information currently stored in the Adj-RIB-In which is identified by the distinguishing attributes of the RIB REFRESH PDU until the refresh cycle has been completed or has been aborted.

The BIS shall accumulate the routeing information contained in all the UPDATE PDUs that are received in a completed refresh cycle. Completion of a refresh cycle is indicated by receipt of a RIB REFRESH PDU with OpCode 3 (RIB Refresh End). Then the BIS shall replace the previous routeing information in the associated Adj-RIB-In with the routeing information that was learned during the refresh cycle.

Abortion of a refresh cycle is indicated by receipt of another RIB REFRESH PDU with OpCode 2 (RIB Refresh Start) before receipt of a RIB REFRESH PDU with OpCode 3 (RIB Refresh End). In this case, any routeing information learned in the time between receipt of the two successive RIB Refresh Starts shall be discarded, and a new refresh cycle (triggered by receipt of the second RIB Refresh Start) shall begin.

If the refreshing BIS receives a new RIB-Refresh-Request while it is in the middle of refresh (after sending RIB-REFRESH PDU with OpCode RIB-Refresh-Start, but before sending RIB-REFRESH PDU with OpCode RIB-Refresh-End), then the current refresh shall be aborted and the new refresh is initiated.

7.11 Path Attributes

An UPDATE PDU that carries an NLRI field also carries a set of *path attributes*. An UPDATE PDU that does not carry any NLRI field shall not carry any path attributes. Path attributes are summarized in Table 3; their encoding is described in clause 6.3.

Table 3. Path Attribute Characteristics

Attribute	Category	Type Code	Length (octets)	Distinguishing
ROUTE_SEPARATOR	well-known discretionary	1	5	No
EXT_INFO	well-known discretionary	2	0	No
RD_PATH	well-known mandatory	3	variable	No
NEXT_HOP	well-known discretionary	4	variable	No
DIST_LIST_INCL	well-known discretionary	5	variable	No
DIST_LIST_EXCL	well-known discretionary	6	variable	No
MULTI-EXIT DISC	optional non-transitive	7	1	No
TRANSIT DELAY	well-known discretionary	8	2	Yes
RESIDUAL ERROR	well-known discretionary	9	4	Yes
EXPENSE	well-known discretionary	10	2	Yes
LOCALLY DEFINED QOS	well-known discretionary	11	variable	Yes
HIERARCHICAL RECORDING	well-known discretionary	12	1	No
RD_HOP_COUNT	well-known mandatory	13	1	No
SECURITY	well-known discretionary	14	variable	Yes
CAPACITY	well-known mandatory	15	1	No
PRIORITY	well-known discretionary	16	1	Yes

7.11.1 Categories of Path Attributes

Path attributes fall into four categories:

a) Well-known mandatory: these attributes must be recognized upon receipt by all BISs, and must be present in every UPDATE PDU
b) Well-known discretionary: these attributes must be recognized upon receipt by all BISs, but are not necessarily present in an UPDATE PDU
c) Optional transitive: these attributes need not be recognized upon receipt by all BISs, and are not necessarily present in an UPDATE PDU. If a given BIS does not recognize an optional transitive attribute, it must pass it on to other BISs
d) Optional non-transitive: these attributes need not be recognized upon receipt by all BISs, and are not necessarily present in an UPDATE PDU. If it does not recognize an optional non-transitive attribute, a BIS shall ignore it and shall not include it in any of its own UPDATE PDUs.

A BIS shall handle optional attributes in the following manner:

a) If a route with an unrecognized optional transitive attribute is received and the route is to be propagated to other BISs, the optional transitive attribute must be propagated with the route, and the Partial bit in the Flag field of the attribute shall be set to 1.

b) If a route with a recognized optional transitive attribute is received and the route is to be propagated to other BISs, the optional transitive attribute may or may not be propagated with the route, according to the definition of the attribute. If the attribute is propagated, then the local BIS shall not modify the value of the PARTIAL bit in the Flag field of the attribute.

c) If a route with an unrecognized optional non-transitive attribute is received, the receiving BIS shall ignore the attribute and shall not propagate that attribute to any other BIS. However, it may propagate the remainder of the route: that is, the route without the unrecognized optional non-transitive attribute.

d) If a route with a recognized optional non-transitive attribute is received and the route is to be propagated to other BISs, the optional transitive attribute may or may not be propagated with the route, according to the definition of the attribute. If the attribute is propagated, then the local BIS shall not modify the value of the PARTIAL bit in the Flag field of the attribute.

BISs shall observe the following rules for attaching and updating the values of optional attributes:

— New optional transitive attributes may be attached to the path information by any BIS in the

path, and that BIS shall then set the PARTIAL bit in the attributes flag of its UPDATE PDU to 1.

— The rules for attaching new non-transitive optional attributes depend on the nature of each specific attribute. The definition of each non-transitive optional attribute specifies such rules.

— Any optional attribute may be updated by any BIS in its path.

7.11.2 Handling of Distinguishing Attributes

Certain well-known discretionary path attributes are classified as Distinguishing Attributes (see clause 5.7), which can be used to discriminate among multiple routes to a destination, based on differences in quality between the routes.

Distinguishing path attributes shall only be created by the BIS that originates the routeing information; they can be updated by any BIS that receives an UPDATE PDU that contains them. The rules for updating each of IDRP's distinguishing attributes are defined in the appropriate subclause of clause 7.12.

A permissible set of distinguishing attributes is defined to be a set that can be derived from information that can be validly encoded in the header of an ISO 8473 NPDU, using the mappings described in 9.2. In turn, a valid RIB-Att (see 5.7) is also a permissible set of distinguishing attributes, which is used to identify the RIB that holds a route characterized by those distinguishing attributes. Therefore, a permissible set of distinguishing attributes and a corresponding valid RIB-Att:

a) Can consist of an empty set (that is, the Empty Distinguishing Attribute)

4 b) Can contain the SECURITY path attribute

4 **NOTE 20:** This distinguishing attributes is derived from the ISO 8473 Security parameter, as described in 8.2.

4 c) Can contain at most one of the following attributes: RESIDUAL ERROR, TRANSIT DELAY, EXPENSE, and LOCALLY DEFINED QOS.

NOTE 21: These distinguishing attributes are derived from the ISO 8473 Quality of Service Maintenance parameter, which can request only one of them (see 8.2).

d) Can include the Priority Distinguishing Attribute.

NOTE 22: This distinguished attribute is derived from the ISO 8473 Priority parameter, as described in 8.2.

e) Can not include any instance of equivalent distinguishing attributes, as defined in 7.11.3.

Therefore, the number of distinguishing attributes that can comprise either a valid RIB-Att or a permissible set of distinguishing attributes is not unbounded: it is limited to at most three.

7.11.3 Equivalent Distinguishing Attributes

IDRP recognizes two categories of distinguishing attribute: type specific, and type-value specific. Certain Distinguishing Attributes are unambiguous by their type —namely, Capacity, Priority, Transit Delay, Expense, and Residual Error. These are called *type specific*.

Others can not be disambiguated based solely on
4 their type, but require knowledge of both type and a
4 subset of the fields that comprise their value—namely,
4 SECURITY and LOCALLY DEFINED QOS. These are called *type-value specific*.

Within IDRP, two instances of Distinguishing Attributes are equivalent each other if either:

a) they are both type specific and they both have the same type, or

b) they are both type-value specific, and they both have the same type and the same value.

In all other cases two instances of Distinguishing Attribute are not equivalent.

7.12 Path Attribute Usage

The usage of each of IDRP's path attributes is described in the following clauses.

1 7.12.1 ROUTE_SEPARATOR

1 ROUTE-SEPARATOR is a well-known mandatory attri-
1 bute. Multiple instances of this attribute may appear
1 in a single UPDATE PDU. The ROUTE_SEPARATOR
1 serves as a delimiter between sets of distinguishing
1 attributes. Each set of distinguishing attributes deter-
1 mines the routeing information base associated with
1 the route. The ROUTE_ID and LOCAL_PREF values
1 for a given route are contained in the
1 ROUTE_SEPARATOR that immediately precedes the
1 set of distinguishing attributes for that route. A BIS
1 shall include a ROUTE_SEPARATOR for each feasible
1 route carried in the UPDATE PDU:

1 — The ROUTE-ID must be unambiguous within the
1 context of the BIS-BIS connection over which the
1 UPDATE PDU is transmitted.
1 — The LOCAL-PREF field is used to detect incon-
1 sistent routeing decisions among a set of BISs
1 that are all located in the same routeing domain.
1 Its value shall be set as follows:

1 a) For UPDATE PDUs sent to adjacent routeing
1 domains, LOCAL-PREF shall contain the value

0; the receiving BIS (in the adjacent RD) shall ignore this field upon receipt.

b) For UPDATE PDUS sent to BISs in the same routeing domain as the local BIS, its value shall be set in accordance with 7.15.1; the receiving BIS (in the same RD) shall use this value to check for internal inconsistencies, in accordance with 7.15.1.

All distinguishing attributes that appear after a given ROUTE_SEPARATOR and before the next ROUTE_SEPARATOR or the end of the BISDPU (whichever occurs first) form a set that determines the RIB-Att associated with the route.

All non-distinguishing path attributes and the NLRI field apply to every route advertised in the UPDATE PDU, regardless of where they appear with respect to the ROUTE_SEPARATOR(s).

The ROUTE_ID associated with a route received from an adjacent BIS bear no functional relationship to the ROUTE_ID that the local BIS will generate if it decides to propagate that route. Similarly, the ROUTE-ID for an aggregated route bears no functional relationship the individual ROUTE-IDs of the routes from which it was constructed.

NOTE 23: The requirements on unambiguity within the context of a given BIS-BIS connection lead to the following observations:

a) Since the ROUTE-ID must be unambiguous within each instance of BIS-BIS communication, a BIS can advertise the same route to different neighbor BISs, using different ROUTE-IDs in each instance of BIS-BIS communications.

b) The ROUTE-IDs associated with routes to be advertised by a BIS (that is, the routes in its Adj-RIBs-Out) bears no relationship to the ROUTE-IDs associated with routes received from other BISs (that is, the routes in the local BIS′s Adj-RIBs-In).

7.12.2 EXT_INFO

EXT_INFO is a well-known discretionary attribute. It shall be recognized upon receipt by all BISs. It shall be included in each UPDATE PDU that reports either an RD_PATH attribute or Network Layer Reachability Information that has been learned by methods not described in this international standard.

The EXT_INFO attribute shall be generated by the RD that originates the associated routeing information. If the EXT_INFO attribute was present in a received UPDATE PDU, then it shall also be included in the UPDATE PDUs of all BISs that choose to propagate this information to other BISs.

NOTE 24: Information obtained from the managed object **INTERNAL-SYSTEMS** or obtained from UPDATE PDUs which do not contain the EXT_INFO attribute has been learned by methods within IDRP′s scope; however, manually configured reachability information for an RD which does not run IDRP is an example of information which is learned by means outside IDRP′s scope.

If a BIS selects a route which has been advertised with the EXT_INFO attribute, it is possible that there may be undetected looping of routeing information. Therefore, it is recommended that distribution of information not learned by the methods of IDRP be tightly controlled. The path attributes DIST_LIST_INCL and DIST_LIST_EXCL afford a convenient method for providing this control. Furthermore, a given RD may also enforce policies which prohibit any of its BISs from selecting routes which have the EXT_INFO attribute associated with them.

7.12.3 RD_PATH

RD_PATH is a well-known mandatory attribute. It shall be present in every UPDATE PDU, and shall be recognized on receipt by all BISs. This attribute consists of a concatenation of path segments that identifies the routeing domains and routeing domain confederations through which this route has passed. The path segments can be RD_SETs, RD_SEQs, ENTRY_SEQs, or ENTRY_SETs.

7.12.3.1 Generating an RD_PATH Attribute

When a BIS originates a route to destinations contained within its own routeing domain or to destinations learned by means outside the protocol (see 7.12.2), it shall examine the information contained in its managed object **RDC-CONFIG** to determine the ordering relationships among all the confederations of which the local routeing domain is a member. The local BIS shall then construct an RD_PATH attribute as follows:

a) If the local routeing domain is a member of one or more confederations, the RD_PATH shall consist of an ENTRY_SEQ segment followed immediately by an RD_SEQ segment. The ENTRY_SEQ shall list the confederations, ordered as follows:

1) If a confederation, RDC-B, is nested within another confederation, RDC-A, then the RDI of RDC-A shall precede that of RDC-B.

2) The RDIs of overlapping confederations shall be listed in increasing order of the RDIs, as long as the order implied by any nesting relationships is maintained. For purposes of ordering, two RDIs are compared octet-by-octet from the left until differing octet values are found. The RDI with the lesser octet value (when treated as an unsigned integer)

is considered to have the lesser RDI value. If there are two RDIs of different lengths, and the leading octets of the longer RDI are exactly the same as the octets of the (complete) shorter RDI, then the shorter RDI is considered to have the lesser value.

The RD_SEQ shall list the RDI of the BIS′s routeing domain.

b) If the local routeing domain is not a member of any confederation, then the RD_PATH contains a single RD_SEQ segment that lists the RDI of the BIS′s routeing domain.

7.12.3.2 Updating a Received RD_PATH Attribute

The local BIS shall update the RD_PATH attribute of a route received from another BIS according to the following rules:

a) If the route was received from a BIS located in the same routeing domain as the local BIS, then the RD_PATH attribute shall not be updated.

b) If the route was received from a BIS located in an adjacent routeing domain, the local BIS shall determine if the route has entered any confederations (see 7.13.3), and it shall examine the information contained in its managed object **RDC-CONFIG** to determine the ordering relationships among all such confederations. The local BIS shall then amend the RD_PATH attribute as follows:

1) If the route has entered any confederations, the BIS shall append a path segment of type ENTRY_SEQ that lists all the newly entered confederations, ordered as follows:

i) If a confederation, RDC-B, is nested within another confederation, RDC-A, then the RDI of RDC-A shall precede that of RDC-B.

ii) The RDIs of overlapping confederations shall be listed in increasing order of the RDIs, as long as the order implied by any nesting relationships is maintained. For purposes of ordering, two RDIs are compared octet-by-octet from the left until differing octet values are found. The RDI with the lesser octet value (when treated as an unsigned integer) is considered to have the lesser RDI value. If there are two RDIs of different lengths, and the leading octets of the longer RDI are exactly the same as the octets of the (complete) shorter RDI, then the shorter RDI is considered to have the lesser value.

The ENTRY_SEQ segment shall be followed immediately by an RD_SEQ segment that lists the RDI of the BIS′s routeing domain.

2) If the route has not entered any confederations, the local BIS shall append a path segment of type RD_SEQ that lists the RDI of the BIS′s routeing domain.

7.12.3.3 Advertising a Route Received from Another BIS

After receiving a route, a BIS will have modified its RD_PATH attribute in accordance with 7.12.3.2; and when a route is generated locally, the BIS will have created an RD_PATH attribute in accordance with 7.12.3.1. If the local BIS selects a route for subsequent advertisement, the RD_PATH attribute of that route shall be amended as follows, based on the confederations which have been exited and on the nesting relationships among confederations of which the local BIS is a member (see managed object **RDC_CONFIG**):

a) If the adjacent BIS to which the route will be advertised can be reached without exiting any confederations, then no modification to the RD_PATH attribute shall be made.

b) If the adjacent BIS to which the route will be advertised can only be reached by exiting one or more confederations, then the local BIS shall check the RD_PATH attribute for the presence of ENTRY_SEQ or ENTRY_SET path segments that contain the RDIs of the exited confederations.

If there is any RDI of an exited confederation which is absent from all ENTRY_SEQ and ENTRY_SET segments, then the route is in error. The local BIS shall send an IDRP ERROR PDU to the BIS that advertised the route, reporting a Misconfigured_RDCs error.

If two confederation, RDC-A and RDC-B, are listed in the same ENTRY_SEQ, and managed object **RDC-Config** indicates that RDC-B is nested within RDC-A, then the RDI of RDC-A shall precede that of RDC-B in the ENTRY_SEQ. If it does not, the local BIS shall send an IDRP ERROR to the BIS that advertised the route, reporting a Misconfigured_RDCs error.

Otherwise, the local BIS shall scan the RD_PATH attribute from the back (right to left, starting at the highest numbered octet) looking for an ENTRY_SEQ or ENTRY_SET path segment that lists an exited confederation. Within a given ENTRY_SET or ENTRY_SEQ segment, the RDI for a given confederation can not be processed until the RDIs for all confederations nested within it have been processed.

For each exited confederation (for example, the confederation whose RDI is ″X″), the advertising

BIS shall then update the RD_PATH of the route as follows:

1) The entry for ″X″ shall be removed from the ENTRY_SEQ or ENTRY_SET segment

2) If ″X″ is the only RDI contained in an ENTRY_SEQ or ENTRY_SET segment of the RD_PATH, then create a path segment of type RD_SEQ that lists ″X″ and insert it in front of the previous entry for ″X″.

3) If the local BIS's routeing domain is a member of other confederations besides ″X″ that are listed in the ENTRY_SEQ or ENTRY_SET segments of the RD_PATH, then:

   i) If ″X″ occurs in an ENTRY_SEQ or ENTRY_SET segment, and ″X″ is nested within none of the other confederations, then create an RD_SET that lists ″X″ and insert it in front of the first ENTRY_SEQ or ENTRY_SET segment that occurs in the RD_PATH.

   ii) If ″X″ occurs in an ENTRY_SEQ and ″X″ is nested within all the other confederations, then create a path segment of type RD_SEQ that lists ″X″ and insert it immediately in front of the previous entry for ″X″

   iii) If ″X″ occurs in an ENTRY_SEQ and ″X″ is nested within some but not all of the other confederations, then create a path segment of type RD_SET that lists ″X″, and insert it immediately after the closest prior entry for any confederation in which ″X″ is nested.

   iv) If ″X″ occurs in an ENTRY_SET and ″X″ is nested within all the other confederations, then create a path segment of type RD_SET that lists ″X″ and insert it immediately in front of the previous entry for ″X″

   v) If ″X″ occurs in an ENTRY_SET and ″X″ is nested within some but not all of the other confederations, then create a path segment of type RD_SET that lists ″X″, and insert it immediately after the the closest prior entry for any confederation in which ″X″ is nested.

   If the procedures call for the insertion of an RD_SET or an RD_SEQ between entries that are contained in a single ENTRY_SET or ENTRY_SEQ, then break the ENTRY_SET or ENTRY_SEQ into two segments of identical type and perform the insertion. For example, if it is necessary to insert RD_SET(X) between entries for ″A″ and ″B″, where ″A″ and ″B″ are contained in ENTRY_SEQ(H,J,A,B,C), the result would be:

   ENTRY_SEQ(H,J,A) RD_SET(X) ENTRY_SEQ(B,C).

   If, after applying these procedures, the ENTRY_SEQ or ENTRY_SET segment in which ″X″ originally occurred is empty, then that path segment shall be deleted, together with any subsequent path segments between itself and the next occurring ENTRY_SEQ or ENTRY_SET segment, or between itself and the end of the RD_PATH attribute if there is no subsequent ENTRY_SEQ or ENTRY_SET segment.

7.12.4 NEXT_HOP

NEXT_HOP is a well-known discretionary attribute. It shall be recognized upon receipt by all BISs.

For purposes of defining the usage rules for this attribute, a subnetwork is transitive with respect to system reachability if all of the following conditions are true:

a) Systems A, B, and C are all attached to the same subnetwork,

b) When A can reach B directly, and B can reach C directly, it follows that A can reach C directly.

Verification of the above conditions should be accomplished by means outside of IDRP. For example, systems located on a common subnetwork could use an ES-IS protocol (such as IS 9542) to ascertain if there is direct reachability between them. Examples of such media are IEEE 802.2 and SMDS.

Consider three BISs attached to a fully connected transitive subnetwork, as shown in Figure 8: A and B share a BIS-BIS connection, B and C share a BIS-BIS connection, but A and C have no BIS-BIS connection between themselves. If C propagates an UPDATE PDU to B, then with respect to the UPDATE PDU advertised by B:

— C is defined to be the *source BIS*
— B is defined to be the *first recipient BIS*
— A is defined to be the *subsequent recipient BIS*.

In terms of these definitions, the following rules apply to the usage of the NEXT_HOP attribute:

a) **Generating the Attribute**

   When a given BIS generates an UPDATE PDU:

   1) It may list its own NET and the SNPAs of subnetworks that connect itself to the remote BIS in the NEXT_HOP attribute of that UPDATE PDU.

   2) It may choose not to include a NEXT_HOP attribute in its UPDATE PDU. When the NEXT_HOP field is not present, it implies that the NET of the BIS that advertises the

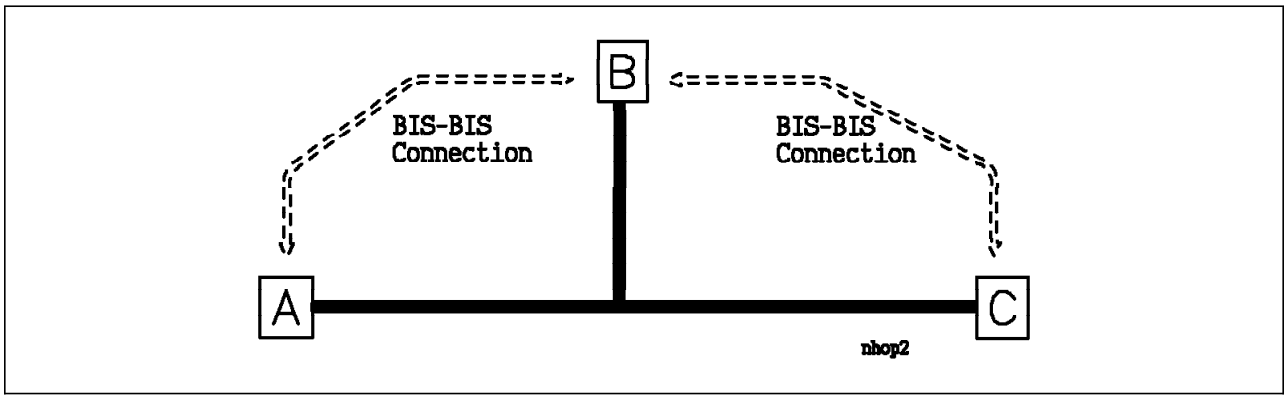

Figure 8. A Transitive Fully Connected Subnetwork

UPDATE PDU should be considered to be the NET of the next-hop BIS.

3) It may set the value of the "IDRP_Server_Allowed" field in accordance with its local policies:

   — If the source BIS wants to allow the first recipient BIS to advertise the source BIS's NET and SNPA to a subsequent recipient BIS, then it shall set this field to X'FF'

   — If the source BIS does not want the first recipient BIS to advertise the source BIS's NET and SNPA, then it shall set this field to any value other than X'FF'.

b) **Advertising Routeing Information**

When a BIS chooses to advertise routeing information learned from an UPDATE PDU:

1) The BIS may choose to list its own NET and the SNPAs of subnetworks that connect itself to the remote BIS in the NEXT_HOP attribute of an UPDATE PDU that propagates the routeing information

2) The BIS may choose not to include a NEXT_HOP attribute in its UPDATE PDU. When the NEXT_HOP field is not present, it implies that the BIS that advertises the UPDATE PDU is also the next-hop BIS.

3) If any condition listed below is not satisfied, then the recipient BIS shall not list the NET and SNPAs of the source BIS in its own UPDATE PDUs. If they are all satisfied, then instead of listing its own NET and SNPAs, the BIS may optionally list the NET and SNPAs of the source BIS (as contained in the UPDATE PDU received from the source BIS) when it propagates the information to a subsequent recipient BIS. The conditions are the following:

   i) The "IDRP_Server_Allowed" field of the UPDATE PDU of the source BIS was equal to X'FF'.

   ii) All three BISs (source, first recipient, and subsequent recipient) are located on a common subnetwork which is full-duplex and is transitive with respect to reachability of all three BISs.

   iii) The managed object **Server** is "true".

   iv) The first recipient and subsequent recipient are located in different routeing domains.

   v) Advertisement of this route to the subsequent recipient BIS does not conflict with any of the path attributes that were contained in the UPDATE PDU from the source BIS. (For example, it may not propagate the UPDATE PDU to a recipient that is listed in the DIST_LIST_EXCL attribute.)

NOTE 25: The following observations should be noted with regard to the rules stated above:

a) The rules do not remove the requirement that there must be a BIS-BIS connections between each pair of BISs located in the same routeing domain.

b) The contents of the NEXT_HOP attribute have no effect upon the contents of the RD_PATH attribute: that is, the RD_PATH attribute will always be used in accordance with clause 7.12.3.

c) If the NET and SNPAs are not available in an UPDATE PDU, then a BIS that receives it must learn them by means outside of this international standard. For example, the value of the NET can be learned from the NUNITDATA.INDICATION, and IS 9542 can be used to associate an SNPA with that NET.

7.12.5 DIST_LIST_INCL

DIST_LIST_INCL is a well-known discretionary attribute. It shall be recognized upon receipt by all BISs. When present, this attribute lists the RDIs of the routeing domains and confederations to which the routeing information may be distributed. Since NPDUs usually flow in a direction opposite to the flow of UPDATE PDUs, DIST_LIST_INCL provides a means for a given RD or confederation to control use of its resources by other RDs.

When a BIS receives an UPDATE PDU that contains the DIST_LIST_INCL attribute, then the receiving BIS shall not redistribute the associated routeing information to any BIS which is not a member of at least one RD or RDC whose RDI is contained in this attribute.

A BIS shall redistribute only that information which has been locally selected as a route, and shall redistribute it only to RDs or RDCs which are both adjacent to it and are included in the distribution list. A BIS may distribute the information to any adjacent BIS that is a member of any routeing domain or confederation whose RDI is contained in DIST_LIST_INCL. A BIS is not required to distribute the routeing information to every RD or RDC whose RDI is listed: for example, it is possible for local policy considerations or the contents of the HIERARCHICAL_RECORDING path attribute to further restrict the set of RDIs to whom the routeing information will actually be redistributed.

If a BIS receives an UPDATE PDU that contains neither the DIST_LIST_INCL nor DIST_LIST_EXCL attributes, then it may distribute the routeing information to all adjacent BISs. Alternatively, the local BIS may also add a DIST_LIST_INCL or DIST_LIST_EXCL attribute, but not both, to the route information.

If the DIST_LIST_INCL attribute is present and has a length of zero octets, then the routeing information may be used locally, but shall not be advertised to any other BIS.

When it originates an UPDATE PDU which describes a route to destinations located in its own routeing domain, a BIS may append the DIST_LIST_INCL attribute, in accordance with its local policies.

If a BIS chooses to advertise a route which was learned from an UPDATE PDU which already contained the DIST_LIST_INCL attribute, the advertising BIS may modify this attribute by pruning the set of RDIs included in the list. If a BIS chooses to prune the set, it shall not delete the RDI of its own RD, nor shall it delete the RDI of any RDC to which it belongs. However, if the reduced list is empty (that is, has a length of zero), then the BIS shall not advertise the routeing information to any BIS located in a different routeing domain.

7.12.5.1 Interaction with HIERARCHICAL_RECORDING

When a given UPDATE PDU contains both the HIERARCHICAL_RECORDING attribute and the DIST_LIST_INCL attribute, the constraints imposed by the HIERARCHICAL_RECORDING, as specified in clause 7.12.12, shall take precedence over those imposed by DIST_LIST_INCL.

7.12.6 DIST_LIST_EXCL

DIST_LIST_EXCL is a well-known discretionary attribute. It shall be recognized upon receipt by all BISs. When present, this attribute lists the RDIs of routeing domains and confederations to which the routeing information may not be distributed. Since NPDUs usually flow in a direction opposite to the flow of UPDATE PDUs, DIST_LIST_EXCL provides a means for a given RD or confederation to control use of its resources by other RDs and RDCs.

When a BIS receives an UPDATE PDU that contains the DIST_LIST_EXCL attribute, then the receiving BIS shall not redistribute the associated routeing information to any BIS located in an RD or RDC whose RDI is included in the list. A BIS shall not distribute the information to any adjacent BIS that is a member of any routeing domain or confederation whose RDI is contained in DIST_LIST_EXCL. Local policy considerations shall not override redistribution of the routeing information as dictated by the DIST_LIST_EXCL attribute.

If a BIS receives an UPDATE PDU that contains neither the DIST_LIST_INCL nor the DIST_LIST_EXCL attributes associated with it, then it may distribute the routeing information to all adjacent BISs. Alternatively, the local BIS may also add a DIST_LIST_INCL or DIST_LIST_EXCL attribute, but not both, to the route information.

If the DIST_LIST_EXCL attribute is absent and the DIST_LIST_INCL attribute is present, then the distribution of the routeing information is controlled by the DIST_LIST_INCL attribute. If the DIST_LIST_EXCL attribute is present and has a length of zero octets, then the routeing information may, in accordance with local policy, be advertised to any other BIS.

When it originates an UPDATE PDU which describes a route to destinations located in its own routeing domain, a BIS may append the DIST_LIST_EXCL attribute, in accordance with its local policies.

If a BIS chooses to advertise a route which was learned from an UPDATE PDU which already con-

tained the DIST_LIST_EXCL attribute, the advertising BIS may modify this attribute by augmenting the set of RDIs included in the list. If a BIS chooses to augment the set, it shall not add the RDI of its own RD, nor shall it add the RDI of any RDC to which it belongs.

7.12.6.1 Interaction with HIERARCHICAL RECORDING

When a given UPDATE PDU contains both the HIERARCHICAL_RECORDING attribute and the DIST_LIST_EXCL attribute, the constraints imposed by DIST_LIST_EXCL, as specified in clause 7.12.6, shall take precedence over those imposed by the HIERARCHICAL_RECORDING attribute.

7.12.7 MULTI-EXIT_DISC

MULTI-EXIT_DISC is an optional non-transitive attribute. If the local BIS's managed object **Multiexit** is "true", the BIS may use the attribute in its path selection algorithm. For example, a routeing domain may choose to implement a policy which mandates that if all other path attributes are equal, the exit point with the lowest value of MULTI-EXIT_DISC should be preferred.

Each BIS that is connected to an adjacent RD by one or more common subnetworks may generate a MULTI-EXIT_DISC attribute for each link connecting itself to an adjacent RD. The value of this attribute is a local matter, which will be determined by the policies of the RD in which the originating BIS is located.

A BIS that generates a value for this attribute may distribute it to all neighboring BISs which are located in adjacent RDs.

If a MULTI-EXIT_DISC attribute is received from a BIS located in an adjacent RD, then the receiving BIS may distribute this attribute to all other BISs located in its own RD. However, the receiving BIS shall not redistribute the attribute to any BISs which are not located within its own RD.

7.12.8 TRANSIT DELAY

TRANSIT DELAY is a well-known discretionary attribute. A BIS shall include this attribute in its UPDATE PDU to indicate that it supports routeing based on the Transit Delay attribute, and that it maintains Adj-RIBs and a Loc-RIB distinguished by this attribute.

The average transit delay associated with the local RD is obtained from the managed object **RDTransitDelay** and represents an average transit delay that would be experienced by SNSDU size of 512 octets while traversing the RD. **RDTransitDelay** is specified in units of 2 ms.

If A BIS advertises a route whose destinations are located in its own RD, then the originating BIS may append the Transit Delay attribute to the route, using the value contained in managed object **RDTransitDelay**.

When a BIS re-distributes a route which has been learned in an UPDATE PDU that contains the Transit Delay attribute, it shall update the value of this attribute before advertising the route to a BIS located in another routeing domain. The updated value shall be computed by adding the value in **RDTransitDelay** to the value of the parameter that was received in the UPDATE PDU's Transit Delay attribute.

If the route is re-distributed to another BIS located in the same RD as the advertising BIS, then the Transit Delay attribute shall not be modified, but shall be distributed with the same value that was present in the received UPDATE PDU.

7.12.9 RESIDUAL ERROR

RESIDUAL ERROR is a well-known discretionary attribute. A BIS shall include this attribute in its UPDATE PDU to indicate that it supports routeing based on the Residual Error attribute, and that it maintains Adj-RIBs and a Loc-RIB distinguished by this attribute.

The value contained in the RESIDUAL ERROR attribute, in *RRE*, and in managed object **RDLRE** is a positive integer in the range from 0 to $2^{32} - 1$; the actual probability of error can be obtained by dividing the value by $2^{32} - 1$.

If A BIS advertises a route whose destinations are located in its own RD, then the originating BIS may append the RESIDUAL ERROR attribute to the route, using the value contained in managed object **RDLRE**. The value of the **RDLRE** is an integer value derived from the average ratio of lost, duplicated, or incorrectly delivered SNSDU's to total SNSDUs transmitted by the SNDCF during a measurement period: this ratio is multiplied by $2^{32} - 1$, and then rounded up to the next higher integer.

When a BIS re-distributes a route which has been learned in an UPDATE PDU that contains the RESIDUAL ERROR attribute, it shall update the value of this attribute before advertising the route to a BIS located in another routeing domain. The updated value is computed by the following formula:

$$K \times (1 - \{(1 - \lceil RRE/K \rceil) \times (1 - \lceil RDLRE/K \rceil)\})$$

where *RRE* is the value of the RESIDUAL ERROR attribute of the received route, *RDLRE* is the value of the average residual error probability associated with the local RD, *K* is the constant $2^{32} - 1$, and the whole expression is rounded up to the nearest integer.

If the route is re-distributed to another BIS located in the same RD as the advertising BIS, then the RESIDUAL ERROR attribute shall not be modified, but shall be distributed with the same value that was present in the received UPDATE PDU.

7.12.10 EXPENSE

EXPENSE is a well-known discretionary attribute. A BIS shall include this attribute in its UPDATE PDU to indicate that it supports routeing based on the Expense attribute, and that it maintains Adj-RIBs and a Loc-RIB distinguished by this attribute. The value of Expense associated with a given routeing domain is contained in managed object **LocExpense**. It is related to monetary cost. Different routeing domains may use different values for this attribute: thus, the attribute must deal in relative monetary costs.

If A BIS advertises a route whose destinations are located in its own RD, then the originating BIS may append the Expense attribute to the route, using the value contained in managed object **LocExpense**.

When a BIS re-distributes a route which has been learned in an UPDATE PDU that contains the Expense attribute, it shall update the value of this attribute before advertising the route to a BIS located in another routeing domain. The updated value is computed by adding the value of the expense contained in managed object **LocExpense** to the value of the EXPENSE attribute of the received route.

If the route is re-distributed to another BIS located in the same RD as the advertising BIS, then the Expense attribute shall not be modified, but shall be distributed with the same value that was present in the received UPDATE PDU.

7.12.11 LOCALLY DEFINED QOS

LOCALLY DEFINED QOS is a well known discretionary attribute that enables a QoS Authority to specify a QoS measurement that is not included in the set of QoS measurements specified in this international standard. The QoS Measurement is identified within the context of the QoS Authority, which is also responsible for specifying its name (QoS Value) and its semantics. A QoS Authority may specify as many QoS measurements as necessary, each distinguished by the QoS Value field.

When a BIS supports a LOCALLY DEFINED QOS measurement, this may be signalled to adjacent BISs in a RIB-Att as specified in clause 6.2. When support of a LOCALLY DEFINED QOS measurement is indicated in a RIB-Att, then the BIS shall support the adj-RIB-ins, adj-RIB-outs and a loc-RIB and a FIB corresponding to this RIB-Att. A BIS may only include a LOCALLY DEFINED QOS measurement in a RIB-Att when its PIB contains the rules necessary to support this measurement.

When a BIS re-distributes a route which has been learned in an UPDATE PDU that contains a LOCALLY DEFINED QOS path attributes, then the new UPDATE PDU shall contain the same QoS Value and NSAP Address Prefix fields in the LOCALLY DEFINED QOS path attribute, and the metric fields (if present) shall be modified in the following way:

a) when the route is advertised to a BIS located in the same RD as the advertising BIS, the metric value shall not be modified

b) when the route is advertised to a BIS located in a different RD from the advertising BIS, the metric value shall be modified according to the rules for that metric. Rules for modifying a given metric value field are defined by the authority responsible for assigning the addresses contained in the ″address″ field of this attribute; such rules shall be specified in the PIB.

7.12.12 HIERARCHICAL RECORDING

The HIERARCHICAL_RECORDING attribute provides an optional means for members of a routeing domain confederation to control transitivity. The transitivity constraints are based upon two factors:

a) the value of the HIERARCHICAL ATTRIBUTE that was contained in a received UPDATE PDU

b) knowledge of whether it is necessary to enter or exit a confederation in order to reach an adjacent RD.

If an RD wishes to support this attribute, then it shall obey the following usage rules:

a) *Destination BIS in a Disjoint RDC*:

The HIERARCHICAL_RECORDING attribute shall not be included in an UPDATE PDU that is transmitted to a BIS that can be reached only by exiting all of the confederations in which the advertising BIS resides.

b) *Destination BIS in Same, Nested, or Overlapping RDC*:

1) If a given BIS chooses to advertise routeing information that it learned from an inbound UPDATE PDU whose HIERARCHICAL_RECORDING attribute was equal to 1, or if it is the originator of the routeing information, it may advertise this information to BISs located in any adjacent routeing domain, as follows:

i) If it is necessary to enter a confederation in order to reach the destination BIS, then the advertising BIS shall include the

HIERARCHICAL RECORDING attribute in its outbound UPDATE PDU, and shall set its value to 0.

ii) If the destination BIS can be reached without entering any confederation, then the advertising BIS shall include the HIERARCHICAL RECORDING attribute in its outbound UPDATE PDU, and shall set its value to 1.

2) If a given BIS chooses to advertise routeing information that it learned from an inbound UPDATE PDU whose HIERARCHICAL_RECORDING attribute was equal to 0, it may advertise that information only to BISs that can be reached without exiting any confederation to which the advertising BIS belongs. The HIERARCHICAL_RECORDING attribute shall be included in the outbound UPDATE PDU, and its value shall be set to 0.

7.12.12.1 Interaction with DIST_LIST_INCL and DIST_LIST_EXCL

When a given UPDATE PDU contains both the HIERARCHICAL_RECORDING attribute and the DIST_LIST_EXCL attribute, the constraints imposed by DIST_LIST_EXCL, as specified in clause 7.12.6, shall take precedence over those imposed by the HIERARCHICAL_RECORDING attribute.

When a given UPDATE PDU contains both the HIERARCHICAL_RECORDING attribute and the DIST_LIST_INCL attribute, the constraints imposed by the HIERARCHICAL_RECORDING, as specified in clause 7.12.12, shall take precedence over those imposed by DIST_LIST_INCL.

7.12.13 RD_HOP_COUNT

This is a well-known mandatory attribute whose usage is as follows:

a) The initial value of this attribute is 0.

b) Before sending an UPDATE PDU to a BIS located in an adjacent routeing domain, a BIS shall increment the value of this attribute by 1, and shall place the result in the RD_HOP_COUNT field of the outbound UPDATE PDU.

c) A BIS shall not increment the value of this attribute when it sends an UPDATE PDU to another BIS located in its own routeing domain.

NOTE 26: ISO 8473 limits the maximum lifetime of an NPDU to 256 counts, and requires each Network entity processing a given NPDU to decrement that NPDU's lifetime by at least 1 count. In the limiting case of one BIS per routeing domain, this implies that a NPDU's lifetime will expire before it can reach the 257th RD. Hence, there is no need to provide an RD_HOP_COUNT greater than 256.

7.12.14 SECURITY

4 SECURITY is a well known discretionary attribute that
4 enables a Security Authority to specify security
4 related information concerning a route. The security
4 related information is identified within the context of
4 the Security Authority, which is also responsible for
4 specifying its semantics. Only one security attribute
4 may be included in each route. When a BIS is able to
4 interpret and act on security related information spec-
4 ified by a given Security Authority, this may be sig-
4 nalled to adjacent BISs in a RIB-Att as specified in
4 clause 6.2. When support of the SECURITY attribute
4 is indicated in a RIB-Att, then the BIS shall support
4 the adj-RIB-ins, adj-RIB-outs and a loc-RIB and a FIB
4 corresponding to this RIB-Att. A BIS may only include
4 a SECURITY distinguishing attribute in a RIB-Att when
4 its PIB contains the rules necessary to interpret and
4 act on the security related information for the identi-
4 fied Security Authority. When a BIS re-distributes a
4 route which has been learned in an UPDATE PDU that
4 contains a SECURITY distinguishing attribute, then the
4 new UPDATE PDU shall contain the same Security
4 Authority identification fields, and the security related
4 information shall be modified according to the rules
4 specified in the PIB. Any such modification may only
4 reduce the protection level indicated, or add addi-
4 tional restrictions on access to the route.

7.12.15 CAPACITY

5 This is a well-known mandatory attribute that is used
5 to denote the traffic handling capacity of the
5 RD_PATH listed in the same UPDATE PDU.

The value of capacity that is associated with a given routeing domain is contained in managed object **Capacity**.

If a BIS advertises a route whose destinations are
5 located in its own routeing domain, then the origi-
5 nating BIS shall include this attribute in its outbound UPDATE PDUs, and its value shall be equal to that of managed object **Capacity**.

5 If a BIS redistributes a route, it shall include the
5 CAPACITY attribute in its outbound UPDATE PDU, and shall reflect the higher of the following two quantities: the value of the CAPACITY attribute contained in the UPDATE PDU that advertised the route, or the value of local managed object **Capacity**.

7.12.16 PRIORITY

This well-known discretionary attribute is a distinguishing attribute, used to indicate the minimum priority value that the BIS will support. It is an unsigned integer in the range from 0 to 14, with higher priority indicated by higher values.

The value of this parameter is the same for all BISs in a given routeing domain, and is equal to the value contained in managed object **Priority**.

If a BIS originates a route to destinations located in its own routeing domain, then the originating BIS may include this attribute in its outbound UPDATE PDUs; if present, its value shall be equal to that of managed object **Priority**.

If a BIS redistributes a route that was advertised with the PRIORITY attribute present, it shall include the PRIORITY attribute in its outbound UPDATE PDU, and shall set its value equal to the higher of the following two quantities: the value of the PRIORITY attribute contained in the UPDATE PDU that advertised the route, or the value of local managed object **Priority**.

7.13 Routeing Domain Confederations

Formation of an RDC is done via a private arrangement between its members without any need for global coordination; the methods for doing so are not within the scope of this international standard.

From the outside, an RDC looks similar to a single routeing domain: for example, it has an identifier which is an RDI. Other RDs can develop policies with respect to the confederation as a whole, as opposed to the individual RDs that are members of the confederation. Confederations can be disjoint, nested, or overlapping (see clause 3.6).

7.13.1 RDC Policies

Each RD within a confederation may have its own set of policies; that is, different RDs in the same confederation can have different policies. For BISs located within the same RD, the methods of clauses 7.15.1 will detect both internal and external routeing inconsistencies.

Since a confederation appears to the external world as if it were an individual RD, IDRP′s loop detection methods will detect routeing information loops through a given confederation. In particular, a route which leaves the confederation and then later re-enters it will be detected as a loop: thus, a route between two RDs that are members of the same confederation will be constrained to remain within that confederation.

7.13.2 RDC Configuration Information

Each BIS that participates in one or more RDCs must be aware of the RDIs of all confederations of which it is a member, and it must know the partial order which prevails between these confederations: that is, it must know the nesting and overlap relationships between all confederations to which it belongs. This information shall be contained in managed object **RDC-Config**, which consists of a list of confederation RDIs and the partial order that prevails among those confederations.

Since RDCs are formed via private arrangement between their members, the partial order of a given confederation is a local matter for that confederation, and bears no relationship to the partial orders that may prevail in different confederations. The RDI of its own routeing domain is contained in managed object **LOCAL-RDI**, as defined in clause 7.3.

7.13.3 Detecting Confederation Boundaries

A given BIS can tell which confederations are common to itself and an adjacent BIS by comparing information obtained from the *Confed-IDs* field of the adjacent BIS′s OPEN PDU with the local BIS′s **RDC-Config** managed object. This knowledge determines when an outbound UPDATE PDU exits a given confederation and when an inbound UPDATE PDU enters a given confederation:

Exiting a Confederation: An UPDATE PDU sent by a given BIS to an adjacent BIS is defined to have exited all those confederations whose RDIs are present in the advertising BIS′s **RDC-Config** managed object but were not reported in the *Confed-IDs* field of the adjacent BIS′s OPEN PDU.

Entering a Confederation: An UPDATE PDU received from an adjacent BIS is defined to have entered all those confederations whose RDIs are present in the receiving BIS′s **RDC-Config** managed object but were not reported in the *Confed-IDs* field of the sending BIS′s OPEN PDU.

7.14 Update-Receive Process

The Update-Receive process is initiated when an UPDATE PDU with no errors is received while the FSM is in the ESTABLISHED state. When this occurs, the BIS shall update the appropriate Adj-RIB-In.

For each feasible route, the Adj-RIB-In is identified by the set of distinguishing path attributes contained between consecutive instances of ROUTE-SEPARATORs or between the last ROUTE_SEPARATOR and the end of the UPDATE PDU. For an unfeasible route, the Adj-RIB-In is identified by

the ROUTE-ID carried in the WITHDRAWN ROUTES field of the UPDATE PDU. The actions to be taken for each route are:

a) If the UPDATE PDU contains a non-empty WITHDRAWN ROUTES field, the previously advertised feasible routes associated with the ROUTE-IDs contained in this field shall be removed from the Adj-RIB-In. The BIS shall run its Decision Process since the previously advertised route is no longer available for use.

b) If the UPDATE PDU contains feasible routes, they shall each be placed in the appropriate Adj-RIB-In, and the following additional actions shall be taken for each route:

   1) If its NLRI and distinguishing attributes are identical to those of a route currently stored in the Adj-RIB-In, then the new route shall replace the older route in the Adj-RIB-In, thus implicitly withdrawing the older route from service. The BIS shall run its Decision Process since the older route is no longer available for use.

   2) If the new route is an overlapping route that is more specific (see 7.16.3.1) than an earlier route contained in the Adj-RIB-In, and the non-distinguishing path attributes of the new route differ from those of the earlier route, the BIS shall run its Decision Process since the more specific route has implicitly made a portion of the less specific route unavailable for use.

   3) If the new route has identical path attributes (both distinguishing and non-distinguishing) to an earlier route contained in the Adj-RIB-In, and is more specific (see 7.16.3.1) than the earlier route, no further actions are necessary.

   4) If a new route has different NLRI from any of the routes currently in the Adj-RIB-In, it shall be placed in the Adj-RIB-In.

   5) If a new route is an overlapping route that is less specific (see 7.16.3.1) than an earlier route in an Adj-RIB-In, the BIS shall place the new route in the appropriate Adj-RIB-In. The earlier, more specific route remains unaffected.

7.15 Information Consistency

Correct operation of this protocol requires that all BISs within a routeing domain apply a consistent set of policies for calculating the degree of preference for an RD-path. An internal inconsistency can occur when there is more than a single path to a particular destination. The hop-by-hop routeing methodology then requires a BIS to select only one of these paths for each distinguishable QOS category. Internal inconsistency arises if different BISs within the same routeing domain make different selections when presented with exactly the same set of routeing information.

7.15.1 Detecting Inconsistencies

The Update-Receive and Update-Send processes of each BIS shall support the LOCAL_PREF field of the ROUTE_SEPARATOR attribute, which is a well-known discretionary attribute. Its value shall be set to zero in UPDATE PDUs transmitted to BISs that are located in adjacent routeing domains. In UPDATE PDUs transmitted to BISs that are located in the same routeing domain as the local BIS, its value shall be set to the degree of preference for the route as computed by the advertising (local) BIS.

A BIS shall detect inconsistent routeing decisions (and, therefore, internal inconsistencies) by calculating a degree of preference for each route carried in an UPDATE PDU that it receives as part of the internal update process (see 7.17.1). If the degree of preference calculated by the local BIS is different from the value carried in the LOCAL_PREF field of the UPDATE PDU's ROUTE_SEPARATOR attribute, the routeing policies of the two BISs have internal inconsistencies. The BIS that detects the inconsistency shall report it to system management.

7.16 Decision Process

The Decision Process selects routes for subsequent advertisement by applying the policies in its Policy Information Base to the routes stored in its Adj-RIBs-In. The output of the Decision Process is the set of routes that will be advertised to adjacent BISs; the selected routes will be stored in the local BIS's Loc-RIBs and Adj-RIBs-Out.

The selection process is formalized by defining a function that takes the attributes of a given route as an argument and returns a non-negative integer denoting the degree of preference for the route. The function that calculates the degree of preference for a given route shall not use as its inputs any of the following: the existence of other routes, the non-existence of other routes, or the path attributes of other routes. Route selection then consists of individual application of the degree of preference function to each feasible route, followed by the choice of the one with the highest degree of preference.

Routes that could form routeing loops must be ignored by the Decision Process. Therefore, any route that was a) received from a BIS located in an adjacent routeing domain and b) contains in its RD_PATH attribute a path segment of type RD_SEQ or RD_SET

1 that contains the RDI of the local routeing domain or
1 any RDC of which the local RD is a member is unfea-
1 sible, and shall be discarded by the Decision Process.

IDRP does not require a universally agreed-upon metric to exist between multiple RDs. Instead, IDRP allows each RD to apply its own set of criteria for route selection, as determined by its local PIB. An example of the syntax and semantics of a routeing policy that could be used to calculate a degree of preference is described in informative Annex K.

The Decision process operates on routes contained in each Adj-RIB-In, and is responsible for:

— selection of routes to be advertised to BISs located in local BIS′s routeing domain
— selection of routes to be advertised to BISs located in adjacent routeing domains
— route aggregation and route information reduction

The Decision process takes place in three distinct phases, each triggered by a different event:

a) Phase 1 is responsible for calculating the degree of preference for each route received from a BIS located in an adjacent routeing domain, and for advertising to the other BISs in the local Routing Domain the routes that have the highest degree of preference for each distinct destination.

b) Phase 2 is invoked on completion of Phase 1. It is responsible for choosing the best route out of all those available for each distinct destination, and for installing each chosen route into the appropriate Loc-RIB.

c) Phase 3 is invoked after the Loc-RIB has been modified. It is responsible for disseminating routes in the Loc-RIB to each adjacent BIS located in an adjacent routeing domain, according to the policies contained in the PIB. Route aggregation, information reduction and the modification of QOS path attributes can optionally be performed within this phase.

7.16.1 Phase 1: Calculation of Degree of Preference

The Phase 1 decision function shall be invoked whenever the local BIS receives an UPDATE PDU from a neighbor BIS that advertises a new route, a replacement route, or a withdrawn route.

The Phase 1 decision function is a separate process which completes when it has no further work to do.

The Phase 1 decision function shall be blocked from running while the Phase 2 decision function for the same RIB-Att is in process.

The Phase 1 decision function shall lock an Adj-RIB-In prior to operating on any route contained within it, and shall unlock it after operating on all new or unfeasible routes contained within it.

For each newly received or replacement feasible route, the local BIS shall compute a degree of preference. Then, the local BIS shall run the internal update process of 7.17.1 to select and advertise the most preferable routes.

A route received from another BIS in the local routeing domain always contains a LOCAL_PREF path attribute. If the value of the LOCAL_PREF path attribute differs from the degree of preference computed above, then an internal inconsistency has occurred, and the local BIS shall report it to system management. If the LOCAL_PREF attribute is not present in the received UPDATE PDU, the local BIS shall issue an IDRP ERROR PDU, reporting Missing_Well-known_Attribute.

7.16.2 Phase 2: Route Selection

The Phase 2 decision function shall be invoked on completion of Phase 1. The Phase 2 function is a separate process which completes when it has no further work to do. The Phase 2 process shall consider all routes that are present in the Adj-RIBs-In, including those received from BISs located in its own routeing domain and those received from BISs located in adjacent routeing domains.

The Phase 2 decision function shall be blocked from running while the Phase 3 decision function is in process. The Phase 2 function shall lock all Adj-RIBs-In with the RIB- Att associated with this instance of the process prior to commencing its function, and shall unlock them on completion.

For each set of destinations for which a feasible route exists in the Adj-RIBs-In identified by the RIB-Att on which this instance of the function operates, the local BIS shall identify the route that has:

a) the highest degree of preference of any route to the same set of destinations, or

b) is the only route to that destination, or

c) is selected as a result of the Phase 2 tie breaking rules specified in 7.16.2.1.

The local BIS shall then install that route in the Loc-RIB, replacing any route to the same destination that is currently held in the Loc-RIB.

4 When RIB-Att includes the priority attribute then more
4 than one route with the same NLRI may be copied to
4 the Loc-RIB as long as their computed preference is
4 less than another such route with the same or lower
4 priority.

When the RIB-Att includes the SECURITY attribute then more than one route with the same NLRI may be copied to the Loc-RIB as long as their computed preference is less than another such route which the applicable PIB Security Policy rules identify as providing equivalent or poorer protection and are usable by the same or more NPDUs.

If a route copied to a Loc-RIB does not have a NEXT_HOP path attribute, then the local BIS shall add that attribute to the entry in the Loc-RIB. The value of the attribute shall be the NET of the adjacent BIS from which the route was received.

Unfeasible routes shall be removed from the Loc-RIB, and corresponding unfeasible routes shall then be removed from the Adj-RIBs-In.

NOTE 27: The decision process should not select a route to destinations located within the local routeing domain if that route would exit the local routeing domain and later re-enter it. Such routes would be rejected by other RDs due to the existence of an RD-loop. Furthermore, the IDRP CLNS Forwarding Process will not forward NPDUs (destined to internal destinations) outside of the local RD, but will instead hand them over to the intra-domain routeing protocol.

7.16.2.1 Breaking Ties (Phase 2)

In its Adj-RIBs-In, a BIS may have several routes to the same destination that have the same degree of preference and also have an equivalent set of distinguishing attributes. The local BIS can select only one of these routes for inclusion in the associated Loc-RIB. The local BIS considers all equally preferable routes, both those received from BISs located in adjacent RDs and those received from other BISs located in the local BIS′s own RD.

Ties shall be broken according to the following rules:

a) If the candidate routes have identical path attributes or differ only in the NEXT_HOP attribute, select the route that was advertised by the BIS in an adjacent routeing domain whose NET has the lowest value. If none of the candidate routes were received from a BIS located in an adjacent routeing domain, select the route that was advertised by the BIS in the local routeing domain whose NET has the lowest value.

b) If all candidate routes contain the MULTI-EXIT_DISC attribute, the candidate routes differ only in their NEXT_HOP and MULTI-EXIT_DISC attributes, and the local BIS′s managed object **Multiexit** is TRUE, select the route that has the lowest value of the MULTI-EXIT_DISC attribute. If multiple candidate routes remain, select the route that was advertised by the BIS whose NET has the lowest value. If the managed object **Multiexit** is false, select the route advertised by the BIS in an adjacent RD whose NET has the lowest value. If none of the candidate routes were received from a BIS located in an adjacent routeing domain, select the route that was advertised by the BIS in the local routeing domain whose NET has the lowest value.

c) If the candidate routes differ in any path attributes other than NEXT_HOP and MULTI-EXIT_DISC, select the route that was advertised by the BIS whose NET has the lowest value. When comparing NETs, if one of the candidate routes was received from a BIS in an adjacent routeing domain, use the NET of the local BIS for the comparison.

For purposes of determining the lowest-valued NET, each binary-encoded NET shall be padded with trailing 0′s in order to bring its length up to 20 octets. The encoded (and possibly padded) NETs shall then be treated as unsigned binary integers.

7.16.3 Phase 3: Route Dissemination

The Phase 3 decision function shall be invoked on completion of Phase 2, or when any of the following events occur:

a) when routes in a Loc-RIB to local destinations have changed

b) when locally generated routes with the EXT_INFO attribute (that is, routes learned by means outside of IDRP) have changed

c) when a new BIS-BIS connection has been established

d) when directed to do so by system management.

The Phase 3 function is a separate process which completes when it has no further work to do.

The Phase 3 Routing Decision function shall be blocked from running while the Phase 2 decision function is in process.

All routes in the Loc-RIB shall be processed into a corresponding entry in the associated Adj-RIBs-Out and FIBs (which are identified by the same RIB-Att), replacing the current entries., The path attributes are updated in accordance with the appropriate subclause of 7.12. Route aggregation and information reduction techniques (see 7.18—7.18.2.3) may optionally be applied. Routes with identical NLRI extracted from the same loc-RIB shall always be aggregated before being copied to an Adj-RIB-Out, and may be aggregated with other routes according to the local Routing Policy. Every FIB shall have an entry for every destination for which a route exists in a Loc-RIB.

A locking scheme should be implemented to prevent simultaneous access to an FIB by both the phase 3 function and forwarding engine. The phase 3 function should first lock an FIB before entering, replacing or deleting an entry, and then unlock that FIB once the operation is complete.

When the updating of the Adj-RIBs-Out and the FIBs is complete, the local BIS shall run the external update process of 7.17.2.

7.16.3.1 Overlapping Routes

A BIS may transmit routes with overlapping NLRI to another BIS. NLRI overlap occurs when a set of destinations are identified in non-matching multiple routes, all of which have the same set of distinguishing attributes. Since IDRP encodes NLRI using prefixes, overlaps will always exhibit subset relationships. A route describing a smaller set of destinations (a longer prefix) is said to be *more specific* than a route describing a larger set of destinations (a shorter prefix); similarly, a route describing a larger set of destinations (a shorter prefix) is said to be *less specific* than a route describing a smaller set of destinations (a longer prefix).

When overlapping routes are present in the same Adj-RIB-In, the more specific routes shall take precedence, in order from most specific to least specific.

This precedence relationship effectively decomposes less specific routes into two parts:

— a set of destinations described only by the less specific route, and
— a set of destinations described by the overlap of the less specific and the more specific routes

The set of destinations described by the overlap represent a portion of the less specific route that is feasible, but is not currently in use. If a more specific route is later withdrawn, the set of destinations described by the overlap will still be reachable using the less specific route.

If a BIS receives overlapping routes from a given neighbor, the Decision Process shall not simultaneously reject the more specific route from neighbor BIS (A) and install A′s less specific route unless the contents of the local BIS′s Adj-RIBs-Out and FIBs insure that NPDUs with destinations listed in the NLRI of A′s more specific route can not be forwarded to the neighbor BIS (A).

Therefore, when presented with overlapping routes from a given neighbor BIS (A), the local BIS could select any of the following options, all of which satisfy the criterion stated above:

a) Install both the less specific and more specific routes received from the given neighbor (A)
b) Install the more specific route received from the given neighbor (A) and reject A′s less specific route
c) Install the non-overlapping part of the less specific and more specific routes received from the given neighbor (A)
d) Install a route formed by the aggregation of the less specific and the more specific route received from the given neighbor (A)
e) Install the less specific route received from the given neighbor (A), and also install another route received from a different neighbor (B) that is simultaneously:
    1) more specific than A′s less specific route, and
    2) less specific than A′s more specific route.
f) Install neither of the routes *received from A*.

7.16.4 Interaction with Update Process

Since the Adj-RIBs-In are used both to receive inbound UPDATE PDUs and to provide input to the Decision Process, care must be taken that their contents are not modified while the Decision Process is running. That is, the input to the Decision Process shall remain stable while a computation is in progress.

Two examples of approaches that could be taken to accomplish this:

a) The Decision Process can signal when it is running. During this time, any incoming UPDATE PDUs will be queued and will not be written into the Adj-RIBs-In. If more UPDATE PDUs arrive than can be fit into the allotted queue, they will be dropped and will not be acknowledged.

b) A BIS can maintain two copies of the Adj-RIBs-In—one used by the Decision Process for its computation (call this the Comp-Adj-RIB) and the other to receive inbound UPDATE PDUs (call this the Holding-Adj-RIB). Each time the Decision begins a new computation, the contents of the Holding-Adj-RIB will be copied to the Comp-Adj-RIB: that is, the a snapshot of the Comp-Adj-RIB is used as the input for the Decision Process. The contents of the Comp-Adj-RIB remain stable until a new computation is begun.

The advantage of the first approach is that it takes less memory; the advantage of the second is that inbound UPDATE PDUs will not be dropped. This international standard does not mandate the use of either of these methods. Any method that guarantees that the input data to the Decision Process will remain stable while a computation is in progress and that is consistent with the conformance requirements of this international standard may be used.

7.17 Update-Send Process

The Update-Send process is responsible for advertising BISPDUs to adjacent BISs. For example, it distributes the routes chosen by the Decision Process to other BISs which may be located in either the same RD or an adjacent RD. Rules for information exchange between BIS located in different routeing domains are given in clause 7.17.2; rules for information exchange between BIS located in the same domain are given in clause 7.17.1.

Distribution of reachability information between a set of BISs, all of which are located in the same routeing domain, is referred to as internal distribution. All BISs located in a single RD must present consistent reachability information to adjacent RDs, thus requiring that they have consistent routeing and policy information among them.

NOTE 28: This requirement on consistency does not preclude an RD from distributing different reachability information to each of its adjacent routeing domains. It does mean that all of a domain's BISs which are attached to a given adjacent domain must provide identical reachability to that domain.

When this protocol is run between BISs located in different routeing domains, the communicating BISs must be located in adjacent routeing domains—that is, they must be attached to a common subnetwork.

7.17.1 Internal Updates

The internal update process is concerned with the distribution of routeing information to BISs located in the local BIS's own routeing domain. Each BIS selects the most preferable route, if any, that it has received from a BIS in an adjacent routeing domain, and distributes that route to every other BIS in its own routeing domain. This process ensures that all BISs in a routeing domain will select the same set of routes.

The following procedures shall be applied separately for each set of Distinguishing Attributes supported by the BIS:

a) When a BIS receives an UPDATE PDU from another BIS located in its own routeing domain, the receiving BIS shall not re-distribute the routeing information contained in that UPDATE PDU to other BISs located in its own routeing domain.

b) When a BIS receives a new feasible route from a BIS in an adjacent routeing domain, it shall advertise that route to all other BISs in its routeing domain by means of an UPDATE PDU if any of the following conditions occur:

   1) the degree of preference assigned to the newly received route by the local BIS is higher than the degrees of preference that the local BIS has assigned to other routes—with the same destinations and the same Distinguishing Attributes—that have been received from BISs in adjacent routeing domains.

   2) there are no other routes—with the same destinations and the same Distinguishing Attributes—that have been received from BISs in adjacent routeing domains.

   3) the newly received route is selected as a result of breaking a tie between several routes that were received from BISs in adjacent routeing domains and that have the highest degree of preference, the same destinations, and the same distinguishing attributes (see 7.17.1.1).

c) When a BIS receives an UPDATE PDU with a non-empty WITHDRAWN ROUTES field, it shall remove from its Adj-RIBs-In all routes whose ROUTE-ID was carried in this field. The BIS shall take the following additional steps:

   1) if the corresponding feasible route had not been previously advertised, then no further action is necessary

   2) if the corresponding feasible route had been previously advertised, then:

      i) if a new route is selected for advertisement that has the same distinguishing attributes and NLRI as the unfeasible route, then the local BIS shall advertise the replacement route

      ii) if a replacement route is not available for advertisement, then the BIS shall include the ROUTE-ID of the unfeasible route in the WITHDRAWN ROUTES field of an UPDATE PDU, and shall send this PDU to each neighbor BIS to whom it had previously advertised the corresponding feasible route.

All feasible routes which are advertised shall be placed in the appropriate Adj-RIB-Out, and all unfeasible routes which are advertised shall be removed from the Adj-RIBs-Out.

7.17.1.1 Breaking Ties (Internal Updates)

If a local BIS has connections to several BISs in adjacent domains, there will be multiple Adj-RIBs-In associated with these neighbors. These Adj-RIBs-In might contain several equally preferable routes to the same destination, all of which have the same set of distinguishing attributes and all of which were advertised by BISs located in adjacent routeing domains. The

local BIS shall select one of these routes, according to the following rules:

a) If all candidate routes contain the MULTI-EXIT_DISC attribute, the candidate routes differ only in their NEXT_HOP and MULTI-EXIT_DISC attributes, and the local BIS′s managed object **Multiexit** is TRUE, select the route that has the lowest value of the MULTI-EXIT_DISC attribute. If multiple candidate routes remain, select the route that was advertised by the BIS whose NET has the lowest value.
b) In all other cases, select the route that was advertised by the BIS whose NET has the lowest value.

For purposes of determining the lowest-valued NET, each binary-encoded NET shall be padded with trailing 0′s in order to bring its length up to 20 octets. The encoded (and possibly padded) NETs shall then be treated as unsigned binary integers.

7.17.2 External Updates

The external update process is concerned with the distribution of routeing information to BISs located in adjacent routeing domains. As part of the Phase 3 route selection process, the BIS has updated its Adj-RIBs-Out and its FIBs. All newly installed routes and all newly unfeasible routes for which there is no replacement route shall be advertised to BISs located in adjacent routeing domains by means of UPDATE PDUs.

Any routes in the Loc-RIB marked as infeasible shall be removed. Changes to the reachable destinations within its own RD shall also be advertised in an UPDATE PDU.

However, advertisement of a given UPDATE PDU shall not violate any distribution constraint imposed by the path attributes of the route contained therein. (For example, DIST_LIST_INCL, DIST_LIST_EXCL, and HIERARCHICAL _RECORDING are attributes that impose distribution constraints on the UPDATE PDU that contains them.)

A BIS shall not propagate an UPDATE PDU that contains a set of distinguishing path attributes that were not listed in the RIB-AttsSet field of the neighbor BIS′s OPEN PDU. If such distinguishing attributes are advertised, it will cause the BIS-BIS connection to be closed, as described in clause 7.20.3.

7.17.3 Controlling Routeing Traffic Overhead

The inter-domain routeing protocol constrains the amount of routeing traffic (that is, BISPDUs) in order to limit both the link bandwidth needed to advertise BISPDUs and the processing power needed by the Decision Process to digest the information contained in the BISPDUs.

7.17.3.1 Frequency of Route Advertisement

The managed object **MinRouteAdvertisementInterval** determines the minimum amount of time that must elapse between advertisements of routes to a particular destination from a single BIS. This rate limiting procedure applies on a per-destination basis, although the value of **MinRouteAdvertisementInterval** is set on a per-BIS basis.

Two UPDATE PDUs sent from a single BIS that advertise feasible routes to some common set of destinations received from BISs in other routeing domains must be separated in time by at least **MinRouteAdvertisementInterval**. For example, any technique that ensures that the separation will be between one and two times the value **MinRouteAdvertisementInterval** is acceptable.

Since fast convergence is needed within an RD, this procedure does not apply for routes received from other BISs in the same routeing domain. To avoid long-lived black holes, the procedure does not apply to the explicit withdrawal of unfeasible routes (that is, routes whose ROUTE_ID is listed in the *Withdrawn Routes* field of an UPDATE PDU).

This procedure does not limit the rate of route selection, but only the rate of route advertisement. If new routes are selected multiple times while awaiting the expiration of **MinRouteAdvertisementInterval**, the last route selected shall be advertised at the end of **MinRouteAdvertisementInterval**.

7.17.3.2 Frequency of Route Origination

The architectural constant **MinRDOriginationInterval** determines the minimum amount of time that must elapse between successive advertisements of UPDATE PDUs that report changes within the advertising BIS′s own routeing domain.

7.17.3.3 Jitter

To minimize the likelihood that the distribution of BISPDUs by a given BIS will contain peaks, jitter should be applied to the timers associated with **MinRouteAdvertisementInterval** and **MinRDOriginationInterval**. A given BIS shall apply the same jitter to each of these quantities regardless of

the destinations to which the updates are being sent: that is, jitter will not be applied on a "per peer" basis.

The amount of jitter to be introduced shall be determined by multiplying the base value in the appropriate managed object by a random factor which is uniformly distributed in the range from 1 – *J* to 1, where *J* is the value of the architectural constant **Jitter**. The result shall be rounded up to the nearest 100 millisecond increment.

An example of a suitable algorithm is shown in informative Annex E, using the architectural constant **jitter**.

7.18 Efficient Organization of Routeing Information

Having selected the routeing information which it will advertise, a BIS may avail itself of several methods to organize this information in an efficient manner.

7.18.1 Information Reduction

Information reduction may imply a reduction in granularity of policy control—after information is collapsed, the same policies will apply to all destinations and paths in the equivalence class.

The Decision Process may optionally reduce the amount of information that it will place in the Adj-RIBs-Out by any of the following methods:

a) **Network Layer Reachability Information:**

Destination NSAP addresses can be represented as NSAP address prefixes. In cases where there is a correspondence between the address structure and the systems under control of a routeing domain administrator, it will be possible to reduce the size of the network layer reachability information that is carried in the UPDATE PDUs.

b) **RD_PATHS:**

RD path information can be represented as ordered RD-SEQUENCES or unordered RD_SETs. RD_SETs are used in the route aggregation algorithm described in 7.18.2. They reduce the size of the RD_PATH information by listing each RDI only once, regardless of how many times it may have appeared in the multiple RD_PATHS that were aggregated.

An RD_SET implies that the destinations listed in the NLRI can be reached through paths that traverse at least some of its constituent RDs. RD_SETs provide sufficient information to avoid routeing loops; however, their use may prune potentially useful paths, since such paths are no longer listed individually as in the form of RD_SEQUENCES. In practice this is not likely to be a problem, since once an NPDU arrives at the edge of a group of RDs, the BIS at that point is likely to have more detailed path information and can distinguish individual paths to destinations.

7.18.2 Aggregating Routeing Information

Aggregation is the process of combining the characteristics of several different routes (or components of a route such as an individual path attribute) in such a way that a single route can be advertised. Aggregation can occur as part of the decision process to reduce the amount of information that will be placed in the Adj-RIBs-Out. For example, at the boundary of a routeing domain confederation an exit BIS can aggregate several intra-confederation routes into a single route that will be advertised externally.

Aggregation reduces the amount of information that BISs must store and exchange with each other. Routes can be aggregated by applying the following procedures separately to path attributes of like type and to the NLRI information.

7.18.2.1 Route Aggregation

Several routes shall not be aggregated into a single route unless the Distinguishing Attributes of each of
1 these route are equivalent, as defined in 7.11.3.

Routes that contain the DIST_LIST_INCL attribute may not be aggregated with routes that contain the DIST_LIST_EXCL attribute.

Routes that have the following attributes shall not be aggregated unless the corresponding attributes of each route are identical: MULTI-EXIT_DISC and NEXT_HOP.

An aggregated route is constructed from one or more component routes. If a component of an aggregated route that has been advertised in an UPDATE PDU becomes unfeasible, then all component routes that comprise the aggregated route, except for the unfeasible component, shall be advertised again, either as separate routes or as a new aggregated route. If the new aggregated route has the same NLRI as the previous aggregated route, then no further actions are necessary, since advertisement of the new aggregated route implicitly marks the old aggregated route as having been withdrawn from use. In all other cases, the original aggregated route must be withdrawn explicitly by means of the *Withdrawn Routes* field of an UPDATE PDU.

7.18.2.2 NLRI Aggregation

The aggregation of the NLRI fields from several routes is straightforward:

— If a shorter NSAP address prefix can be used to represent the NSAPs (and only those NSAPs) listed in several individual NSAP address prefixes, this may be done. However, a shorter NSAP prefix which would be associated with more NSAPs than were associated with the individual prefixes being aggregated shall not be used.

— Individual NSAP address prefixes from the routes whose NLRI is to be aggregated can be listed individually in a single NLRI field.

7.18.2.3 Path Attribute Aggregation

Path attributes that have different type codes can not be aggregated together. Path attributes of the same type code may be aggregated, according to the following rules:

ROUTE-ID attributes: When several routes are aggregated, an unambiguous ROUTE-ID shall be assigned to the aggregated route, in accordance with 7.12.1.

EXT_INFO attributes: If at least one route among routes that are aggregated has the EXT_INFO attribute, then the aggregated route must have the EXT_INFO attribute as well.

RD_PATH attributes: The individual RD_PATH attributes from which the aggregated RD_PATH attribute will be constructed are called the *component attributes*, and the ENTRY_SEQ and ENTRY_SET path segments contain the RDIs of confederations that have been entered but not yet exited. If the RDIs of all such confederations appear in the same relative order of entry in every component route, then aggregation may be performed without pre-processing the component routes. If they appear in different orders of entry in the component routes, then the pre-processing step outlined below must be performed in order to create the same order of entry in every component route before applying the aggregation procedures.

If routes to be aggregated have identical RD_PATH attributes, then the aggregated route has the same RD_PATH attribute as each individual route, and no further processing is necessary.

Pre-processing to Attain Identical Order of Entry: Apply the following procedure to each component route individually. Replace all path segments, from the first ENTRY_SET or ENTRY_SEQ segment to the last path segment, inclusive, with a path segment of type ENTRY_SET followed by a path segment of type RD_SET:

— The path segment of type ENTRY_SET shall contain the union of the all the RDIs listed in the individual ENTRY_SET and ENTRY_SEQ segments. The RDIs must be listed in the same order in each component route. The specific ordering algorithm is left as a local matter, but it shall guarantee that the RDI of a given confederation does not precede the RDI of any confederation within which it it is nested.

— The path segment of type RD_SET shall contain the union of the RDIs contained in all RD_SETs and RD_SEQs that appear after the first ENTRY_SET or ENTRY_SEQ of the component route.

Aggregation Procedures: For purposes of this procedure, a path segment that lists multiple RDIs shall be treated as if it were multiple consecutive path segments, where each path segment lists a single RDI and the order of appearance of RDIs is maintained. For example, a path segment that listed RDIs X, Y, and Z (in that order) is treated as if it were a path segment listing X, followed by a path segment listing Y, followed by a path segment listing Z. If all the RD_PATH attributes of all component routes are identical, the aggregated path attribute is equal to the original RD_PATH attribute.

The main procedure of 7.18.2.3.1 calls the subroutine of 7.18.2.3.2 for aggregating RD_PATH attributes that contain no ENTRY_SEQs or ENTRY_SETs (generically called an ″Entry Marker″). In effect, the main procedure applies the subroutine to all segments that are located between Entry Markers, between an Entry Marker and the end of a component attribute, or between the start of a component attribute and its first Entry Marker.

The main procedure is described in 7.18.2.3.1, and the subroutine is described in 7.18.2.3.2.

DIST_LIST_INCL attributes: The DIST_LIST_INCL attribute of the aggregated route is formed by the set intersection of the RDIs listed in the DIST_LIST_INCL attributes of the individual routes. If the set intersection consists of an empty set, then the routes shall not be aggregated.

DIST_LIST_EXCL attributes: The DIST_LIST_EXCL attribute of the aggregated route is formed by the set union of the RDIs listed in the DIST_LIST_EXCL attribute of the individual routes. A route that contains no DIST_LIST_EXCL attribute is treated as if it contained a DIST_LIST_EXCL attribute that lists no RDIs.

TRANSIT DELAY attributes: The value of the Transit Delay attribute of the aggregated route is set to the maximum value of the Transit Delay attribute of the individual routes that are aggregated.

RESIDUAL ERROR attributes: The value of the Residual Error attribute of the aggregated route is set to the maximum value of the Residual Error attribute of the individual routes that are aggregated.

EXPENSE attributes: The value of the Expense attribute of the aggregated route is set to the maximum value of the Expense attribute of the individual routes that are aggregated.

LOCALLY DEFINED QOS attributes: Routes that contain the LOCALLY SPECIFIED QOS attribute shall only be aggregated if their LOCALLY SPECIFIED QOS attributes have an identical NSAP Address Prefix field and an identical QoS Value field.

The rules for determining the value of the metric field of each such LOCALLY SPECIFIED QOS attribute of the aggregated route are specific for each QoS Value and are specified by the responsible QoS Authority. They shall be held in the PIB. If no suitable rule exists in the PIB then the routes shall not be aggregated.

HIERARCHICAL RECORDING attributes: If any of the routes to be aggregated contains the HIERARCHICAL_RECORDING attribute, the aggregated route shall also contain a HIERARCHICAL_RECORDING attribute:

— If the routes to be aggregated contain different values for this attribute, then it shall be set to a value of 0 in the aggregated route

— If all routes to be aggregated contain the same value for this attribute, then it shall be set to that value in the aggregated route.

RD_HOP COUNT attribute: The value of the RD_HOP_COUNT of the aggregated route shall be set equal to the largest RD_HOP_COUNT that was contained in the routes being aggregated.

SECURITY attribute: Routes that contain the SECURITY attribute shall only be aggregated if their SECURITY attributes identify the same Security Authority.

The rules for determining the value of the SECURITY attribute of the aggregated route are specified by the responsible Security Authority. They shall be held in the PIB. If no suitable rule exists in the PIB then the routes shall not be aggregated.

PRIORITY attributes: The value of the PRIORITY attribute of the aggregated route shall be equal to the maximum value of the values of the PRIORITY attributes of the routes being aggregated, unless the NLRI of component routes are identical. In this case, the value of the PRIORITY attribute of the aggregated route shall be the minimum value of the values of the PRIORITY attributes of the routes being aggregated.

CAPACITY Attributes: The value of the CAPACITY attribute of the aggregated route shall be equal to the smallest integer value contained in the CAPACITY fields of the routes being aggregated.

7.18.2.3.1 Main Procedure for RD_PATH Aggregation: This procedure is used to aggregate the RD_PATH attributes of component routes:

a) Set the aggregated RD_PATH to ″empty″.

b) Scanning from the back of each non-empty component attribute, locate the first Entry Marker. If the type of marker in any component route is ENTRY_SET, then change the type of the corresponding Entry Marker in all component attributes to ENTRY_SET.

c) If no Entry Marker is found, apply the subroutine for aggregating RD_PATHs with no Entry Markers (see 7.18.2.3.2), and prepend the result to the aggregated RD_PATH attribute.

d) If a Entry Marker is found, prepend the following to the aggregated RD_PATH attribute, in the order indicated: the located Entry Marker, followed immediately by the path segments obtained by applying the subroutine for aggregation of RD_PATHs with no Entry Markers (see 7.18.2.3.2) to the path segments that follow the located Entry Marker in each component attribute. If a component attribute has no path segments following the located Entry Marker, pass it to the subroutine as an empty set.

e) Delete from each component attribute all the path segments that were appended to the aggregated attribute in steps c or d.

f) Repeat steps b through e until every component attribute is empty.

If there are consecutive path segments of the same type, they shall be combined into a single path segment of the same type.

7.18.2.3.2 Aggregating Routes with No Entry Markers: The subroutine for aggregating RD_PATH attributes with no entry markers is as follows:

a) Set the aggregated RD_PATH to ″empty″.

b) Scanning from the back of each component attribute, locate the first identical longest sequence of path segments that occurs in every component attribute, including any that are empty.

NOTE 29: It will not be possible to find an identical sequence in every component attribute if one or more of them are empty.

c) If there is no identical sequence, form a path segment of type RD_SET that contains every RDI in every non-empty component attribute.

Prepend this list to the aggregated RD_PATH attribute.

d) If the identical sequence is the final sequence of every component attribute, prepend it to the aggregated route.

e) If the identical sequence is not the final sequence of every component attribute, form a path segment of type RD_SET that lists every RDI that occurs between the end of the identical sequence and the end of each non-empty component attribute. Prepend this list to the aggregated RD_PATH attribute.

f) Delete from each component attribute all path segments that were added to the aggregated RD_PATH attribute in step c, d, or e.

g) If, after the deletions in step f have been made, an RDI is present in both the aggregated RD_PATH attribute and in any of the component attributes, then the accumulated RD_PATH attribute shall be replaced by a single path segment of type RD_SET that lists every RDI that was present in the component routes that were the input to this subroutine (before any deletions were made), and the subroutine terminates. Otherwise, repeat steps b through f until every component attribute is empty.

7.19 Maintenance of the Forwarding Information Bases

As summarized in Table 1, the Forwarding Information Bases contain the following information for a given RIB-Att (set of distinguishing attributes) and set of destinations (NLRI):

a) the NET of the next-hop BIS,

b) the local SNPA used by the local BIS to forward traffic to the next-hop BIS,

c) the minimum priority supported by this subnetwork hop, if the FIB′s RIB-Att contains the PRIORITY attribute

d) security related information associated with this subnetwork hop, if the FIB′s RIB-Att contains the SECURITY attribute

e) if available, the SNPA in the next-hop BIS to which NPDUs will be forwarded.

f) the QoS metric value if the FIB′s RIB-Att contains the RESIDUAL ERROR, TRANSIT DELAY, EXPENSE, or LOCALLY DEFINED QOS attribute.

The RIB-Att of the Loc-RIB which contains a route is also the RIB-Att of the corresponding FIB; the NLRI for the associated FIB is the same as the NLRI of the corresponding route that is stored in the Loc-RIB. Therefore, the destination address of an incoming NPDU and its NPDU-derived distinguishing attributes (see 8.3) allow a BIS to determine the FIB which contains the forwarding information to be used for this NPDU.

The forwarding information consists of three parts.

a) **Net of Next-hop BIS**: For each route in the Loc-RIB, the next-hop BIS has been determined, and is carried as a tag, as described in 7.16.2. The same tag is then carried over into the corresponding entry in the FIB. This information is always present.

b) **Output SNPA**: The SNPA that will be used by the local BIS for forwarding traffic to the destinations identified in the NLRI field of the FIB is established locally, and is one of the SNPAs identified in managed object **LocalSNPAs**.

c) **Input SNPA**: The SNPA that will be used by the remote BIS to receive traffic that is the NEXT_HOP attribute of the corresponding route stored in the Loc-RIB. If the NEXT-HOP attribute contains an empty SNPA list, or if the NEXT_HOP attribute itself is not included in the route, then the Input SNPA field in the FIB will be empty.

d) **priority**: The minimum priority that an NPDU shall have for it to be forwarded over this subnetwork hop. If omitted, then the NPDU may have any priority.

e) **security related information**: identifies the protection available over the subnetwork hop and the NPDUs that may use it with reference to a known Security Policy.

f) **QoS Metric**: provides the value of the route′s QoS metric for the corresponding QoS distinguishing path attribute, for use in additional matching rules during the NPDU forwarding process.

7.20 Error Handling for BISPDUs

This section describes actions to be taken when errors are detected while processing BISPDUs.

7.20.1 BISPDU Header Error Handling

If BIS-BIS connection was established using authentication code 2 (checksum plus authentication) and the validation pattern in the BISPDU header does not match the locally computed pattern, then the BISPDU shall be discarded without any further actions.

If any of the following error conditions are detected, the BISPDU shall be discarded, and the appropriate error event shall be logged by the receiving BIS:

a) Length field of a PDU header less than 30 octets or greater than the Segment Size specified by the remote system's OPEN PDU,

b) Length field of an OPEN PDU less than minimum length of an OPEN_ PDU

c) Length field of an UPDATE PDU less than minimum length of an UPDATE PDU

d) Length field of KEEPALIVE PDU not equal to 30

e) Length of an IDRP ERROR PDU less than the minimum length of 32

f) Length of a CEASE PDU less than the minimum length of 30

g) The BIS-BIS connection was established using authentication code 1 (checksum without authentication) and the validation pattern in the BISPDU header does not match the locally computed pattern

h) Type field in the BISPDU is not recognized

7.20.2 OPEN PDU Error Handling

The following errors detected while processing the OPEN PDU shall be indicated by sending an IDRP ERROR PDU with error code OPEN_PDU_Error. The error subcode of the IDRP ERROR PDU shall elaborate on the specific nature of the error.

a) If the version number of the received OPEN PDU is not supported, then the error subcode of the IDRP ERROR PDU shall be set to Unsupported_Version_Number. The Data field of the IDRP ERROR PDU is a 2-octet unsigned integer, which indicates the highest supported version number less than the version of the remote BIS peer's bid (as indicated in the received OPEN PDU).

b) If the Maximum PDU Size field of the OPEN PDU is less than **MinPDULength** octets, the error subcode of the IDRP ERROR PDU is set to Bad_Maximum PDU_Size. The Data field of the IDRP ERROR PDU is a 2 octet unsigned integer which contains the erroneous Maximum PDU Size field.

c) If the Routeing Domain Identifier field of the OPEN PDU is not the expected one, the error subcode of the IDRP ERROR PDU is set to Bad_Peer_RD. The expected values of the Routeing Domain Identifier may be obtained by means outside the scope of this protocol (usually it is a configuration parameter). The value of the erroneous RDI is returned in the Data field of the IDRP ERROR PDU, encoded as a <length, RDI> pair. "Length" is a one octet field containing a positive integer that gives the number of octets used for the following "RDI" field.

d) If a BIS receives an OPEN PDU from a BIS located in the same RD, and the RIB-AttsSet field contained in that PDU is different from the receiving BIS's managed object **RIB-AttsSet**, then the error subcode of the IDRP ERROR PDU shall be set to Bad-RIB-AttsSet.

e) If the value of the Authentication Code field of the OPEN PDU is any value other than 1 or 2, the error subcode of the IDRP ERROR PDU is set to Unsupported_Authentication_Code.

f) If a given BIS receives an OPEN PDU from another BIS located in the same routeing domain, then the RDIs reported in the Confed-IDs field of the OPEN PDU (received from the remote BIS) should match the Confed-IDs of the local BIS. If they do not match exactly, then an IDRP ERROR PDU shall be issued, indicating an OPEN PDU error with an error subcode of RDC_Mismatch. The data field of the IDRP ERROR PDU shall report the offending Confed-IDs field from the rejected OPEN PDU.

7.20.3 UPDATE PDU Error Handling

All errors detected while processing the UPDATE PDU are indicated by sending an IDRP ERROR PDU with error code UPDATE_PDU_Error. The error subcode of the IDRP ERROR PDU elaborates on the specific nature of the error.

a) If the Total Attribute Length is inconsistent with the Length field of the PDU header, then the error subcode of the IDRP ERROR PDU shall be set to Malformed_Attribute_List. No further processing shall be done and all information in the UPDATE PDU shall be discarded.

b) If any recognized attribute has attribute flags that conflict with the attribute type code, then the error subcode of the IDRP ERROR PDU shall be set to Attribute_Flags_Error. The Data field of the IDRP ERROR PDU shall contain the incorrect attribute (type, length and value). No further processing shall be done, and all information in the UPDATE PDU shall be discarded.

c) If any recognized attribute has a length that conflicts with the expected length (based on the attribute type code), then the error subcode of the IDRP ERROR PDU shall be set to Attribute_Length_Error. The Data field of the IDRP ERROR PDU contains the incorrect attribute (type, length and value). No further processing shall be done, and all information in the UPDATE PDU shall be discarded.

d) If any of the mandatory well-known attributes are not present, then the error subcode of the IDRP ERROR PDU shall be set to Missing_Well-known_Attribute. The Data field of

the IDRP ERROR PDU contains the attribute type code of the missing well-known attribute.

e) If any well-known attribute (so designated by the attribute flags) is not recognized, then the error subcode of the IDRP ERROR PDU shall be set to Unrecognized_Well-known_Attribute. The Data field of the IDRP ERROR PDU shall report the unrecognized attribute (type, length and value). In both cases no further processing shall be done, and all information in the UPDATE PDU shall be discarded.

f) If the NEXT_HOP attribute field is invalid, then the error subcode of the IDRP ERROR PDU shall be set to Invalid_NEXT_HOP_Attribute. The Data field of the IDRP ERROR PDU contains the incorrect attribute (type, length and value). No further processing shall be done and all information in the UPDATE PDU shall be discarded.

g) The sequence of RD path segments shall be checked for RD loops. RD loop detection shall be done by scanning the complete list of RD path segments (as specified in the RD_PATH attribute) and checking that each RDI in this list occurs only once. If an RD loop is detected, then the error subcode of the IDRP ERROR PDU shall be set to RD_Routeing_Loop.

The data field of the IDRP ERROR PDU shall report the first RDI that indicated a loop. This RDI shall be followed immediately by the complete RD_PATH attribute. The encoding shall be: length, RDI, Offending RD_PATH attribute>, where:

— ″length″ is a one octet field that gives the length of the in octets of the immediately following RDI field
— ″RDI″ is the RDI that was detected as creating the loop
— RD_PATH is the octet string that encoded the value field of the offending RD_PATH attribute in the received UPDATE PDU (see 6.3.

No further processing shall be done, and all information in the UPDATE PDU shall be discarded.

h) If any non-empty set of distinguishing attributes for any route advertised in an UPDATE PDU received from a BIS located in a different routeing domain does not match any of the RIB-Atts that the local (receiving) BIS had advertised to that neighbor in the RIB-AttsSet field of its OPEN PDU, then the receiving BIS shall send an IDRP Error PDU that reports an error subcode of Malformed_Attribute_List. All information in the UPDATE PDU shall be discarded, and no further processing shall be done.

i) If the UPDATE PDU contains both the DIST_LIST_INCL and the DIST_LIST_EXCL attributes, then the error subcode of the IDRP ERROR PDU shall be set to Malformed_Attribute_List. No further processing shall be done, and all information in the UPDATE PDU shall be discarded.

j) If a BIS receives an UPDATE PDU that has a DIST_LIST_EXCL attribute with the RDI of the receiving BIS or the RDI of any RDC to which the BIS belongs, the subcode of the IDRP ERROR PDU shall be set to Malformed_Attribute_List. The Data field of the IDRP ERROR PDU shall report the incorrect attribute (type, length and value). No further processing shall be done, and all information in the UPDATE PDU shall be discarded.

k) If a BIS receives an UPDATE PDU that has a DIST_LIST_INCL attribute without the RDI of the receiving BIS or the RDI of any RDC to which the BIS belongs, the subcode of the IDRP ERROR PDU shall be set to Malformed_Attribute_List. The Data field of the IDRP ERROR PDU shall report the incorrect attribute (type, length and value). No further processing shall be done, and all information in the UPDATE PDU shall be discarded.

NOTE 30: It is permissible for an UPDATE PDU to contain neither the DIST_LIST_INCL nor the DIST_LIST_EXCL attributes. According to clause 7.12.5, the absence of both the DIST_LIST_INCL and DIST_LIST_EXCL attributes implies that all RDs and all RDCs may receive the routeing information.

l) If the length of the NLRI is inconsistent with the Length field of the PDU header, then the error subcode of the IDRP ERROR PDU shall be set to Malformed_NLRI. No further processing shall be done, and all information in the UPDATE PDU shall be discarded.

m) If an optional attribute is recognized, then the value of this attribute shall be checked. If an error is detected, the attribute shall be discarded, and the error subcode of the IDRP ERROR PDU shall be set to Optional_Attribute_Error. The Data field of the IDRP ERROR PDU shall report the attribute (type, length and value). No further processing shall be done, and all information in the UPDATE PDU shall be discarded.

n) If RDCs are supported and any of the error conditions noted in clause 7.12.3.3 occur, no further processing of the UPDATE PDU shall be done, all information in the UPDATE PDU shall be discarded, and the error code of the NOTIFICATION PDU shall be set to Misconfigured_RDCs.

o) If a route carried in an UPDATE PDU contains more than a single instance of a distinguishing path attribute, then the error subcode shall be set to Malformed_Attribute_List. No further processing shall be done, and all information in the UPDATE PDU shall be discarded.

NOTE 31: This error condition refers to duplicated attributes *within a single route*. It is not an error if a distinguishing attribute of the same type appears in several of the routes carried in an UPDATE PDU that advertises multiple routes.

p) If an UPDATE PDU contains more than one instance of a non-distinguishing path attribute of the same type, except for ROUTE_SEPARATOR, the BIS shall send an IDRP ERROR PDU with error subcode Duplicated_Attributes. The data field of the IDRP ERROR PDU shall list the type codes of all such duplicated attributes.

q) If the RD_PATH attribute contains an illegal segment type, the BIS shall send an IDRP ERROR PDU, with error subcode Illegal_RD_Path_Segment. The data field of the IDRP ERROR PDU shall reproduce the encoding of the offending segment of the RD_PATH attribute, as it appeared in the received UPDATE PDU.

7.20.4 IDRP ERROR PDU Error Handling

If a BIS receives an IDRP ERROR PDU with a correct validation pattern but which contains an unrecognized error code or error subcode, the local BIS shall close the connection as described in clause 7.6.2.

NOTE 32: Any error (such as unrecognized Error Code or Error Subcode, or an incorrect Length field in the PDU header) should be logged locally and brought to the attention of the administration of the peer. The means to do this are, however, outside the scope of this protocol.

7.20.5 Hold Timer Expired Error Handling

If a system does not receive successive KEEPALIVE or UPDATE PDUs within the period specified in the Hold Time field of the OPEN PDU, then an IDRP ERROR PDU with error code Hold_Timer_Expired shall be sent, and the FSM for the associated BIS-BIS connection shall enter the CLOSE-WAIT state.

7.20.6 KEEPALIVE PDU Error Handling

The KEEPALIVE PDU consists of only the BISPDU Header. Error conditions are handled according to 7.20.1.

7.20.7 CEASE PDU Error Handling

The CEASE PDU consists of only the BISPDU Header. Error conditions are handled according to 7.20.1

7.20.8 RIB REFRESH PDU Error Handling

If any of the following error conditions are detected, the BIS shall issue an IDRP ERROR PDU with the following error indications:

a) Invalid OpCode not in Range 1 to 3: indicate RIB REFRESH error with error subcode ″Invalid OpCode″

b) Receipt of an OpCode 3 (RIB Refresh End) without prior receipt of OpCode 2 (Rib Refresh Start): indicate FSM Error

c) Receipt of an unsupported RIB-Att in the Rib-Atts variable length field in the RIB REFRESH PDU for a RIB Refresh Start OpCode: indicate RIB REFRESH error with error subcode ″Unsupported RIB-Atts″

8. Forwarding Process for CLNS

The forwarding process for CLNS operation is driven by the header information carried in an ISO 8473 NPDU:

a) If the NPDU contains an ISO 8473 complete source route parameter, then further forwarding of this NPDU shall be handled by the ISO 8473 protocol, not by the mechanisms defined in this international standard.

b) If the NPDU contains an ISO 8473 partial source route parameter, the NPDU shall be forwarded on a path to the next system listed in the partial source route parameter.

c) If the NPDU does not contain an ISO 8473 source route parameter, the NPDU shall be forwarded on a path to the system listed in the destination address field of the NPDU.

Having determined the system to which a path is needed, the BIS shall proceed as follows:

a) If the destination system is located in its own RD, the local BIS shall proceed as defined in clause 8.1.

b) If the destination system is located in a different RD, the local BIS shall perform the following actions:

1) It shall determine the *NPDU-Derived Distinguishing Attributes* of the NPDU, according to clause 8.2.

2) It shall next apply the procedures of 8.3 to determine if the NPDU-derived Distinguishing Attributes match any of the FIB-Atts of the Forwarding Information Bases of the local BIS:

i) If the NPDU-derived Distinguishing Attributes match the FIB-Att of a local FIB, then the NPDU shall select that FIB and shall forward the NPDU using the methods of clause 8.4.

ii) Otherwise, perform the ISO 8473 "Discard PDU Function" and generate an ER PDU with the parameter value set to "Unsupported Option not Specified".

If the underlying data protocol permits the modification or removal of the QOS or Priority parameters of the data PDU, then the BIS may modify such information appropriately and forward the data PDU according to a fallback FIB. Otherwise, the BIS shall discard the data PDU if changing the QOS or Priority parameter as described above would violate conformance to the underlying data protocol as seen by systems other than the participating BIS.

8.1 Forwarding to Internal Destinations

If the destination address of an incoming NPDU depicts a system located within the routeing domain of the receiving BIS, then it shall forward that NPDU to any of the ISs listed in managed object **INTRA-IS**. That is, any further forwarding of the NPDU is the responsibility of the intra-domain routeing protocol.

8.2 Determining the NPDU-derived Distinguishing Attributes

As the first step in forwarding an NPDU to a destination located in another routeing domain, the receiving BIS shall determine the *NPDU-derived Distinguishing Attributes* of the incoming ISO 8473 NPDU. This determination shall be based on an examination of the priority parameter, security parameter, and QOS maintenance parameter in the NPDU's header:

— The 8473 priority parameter corresponds to the PRIORITY path attribute.

— The first two bits of the ISO 8473 security parameter are decoded:

- if they equal '11', then the parameter identifies a globally unique and unambiguous security level (see ISO 8473, clause 7.5.3.3)
- if they equal '01' then the responsible Security Authority is identified by the NPDU's source NSAP Address
- if they equal '10' then the responsible Security Authority is identified by the NPDU's destination NSAP Address.

The corresponding NPDU-Derived Distinguishing attribute is then a SECURITY attribute identifying the same Security Authority.

— The first two bits of the ISO 8473 QoS Maintenance parameter are decoded:

a) If they equal '01' then the responsible QoS Authority is indicated by the source NSAP Address, and the NPDU-Derived Distinguishing attribute is determined using the remaining octets of the QoS Maintenance parameter and by applying the rules specified by the QoS Authority and contained in the PIB for selection of the NPDU-Derived Distinguishing attribute. If no such rules exist then no NPDU-Derived Distinguishing attribute shall be associated with this QoS Maintenance parameter.

b) If they equal '10' then the responsible QoS Authority is indicated by the destination NSAP Address, and the NPDU-Derived Distinguishing attribute is determined using the remaining octets of the QoS Maintenance parameter and by applying the rules specified by the QoS Authority and contained in the PIB for selection of the NPDU-Derived Distinguishing attribute. If no such rules exist then no NPDU-Derived Distinguishing attribute shall be associated with this QoS Maintenance parameter.

c) If they equal '11' then the NPDU-Derived Distinguishing attribute is as shown in Table 4.

If examination of the 8473 header shows that no NPDU-Derived Distinguished Attributes are present, then the NPDU shall be associated with the Empty Distinguishing Attribute.

8.3 Matching RIB-Att to NPDU-derived Distinguishing Attributes

Within the BIS, each of its FIB(s) has an unambiguous RIB-Att (see clause 7.10.1) which is constructed from the set of Distinguishing Attributes that the local BIS supports. The set of NPDU-derived Distinguishing Attributes matches a given RIB-Att (which is itself a set of Distinguishing Attributes) when all of the following conditions are satisfied:

a) Both sets contain the same number of attributes.

b) Each instance of a type-specific attribute in the NPDU-derived Distinguishing Attributes must have an equivalent instance in the FIB-Att. The type-specific path attributes supported by IDRP are:

— Transit delay
— Residual error
— Expense
— Priority

c) Each instance of a type-value specific attribute in the NPDU-Derived Distinguishing Attributes has a

Table 4. NPDU-Derived Attribute Set. Some NPDU-derived Distinguishing attributes are by examining the QOS Maintenance Parameter octet for 1 or 0 in the bit positions shown below. The symbol "-" indicates that the corresponding bit does not enter into the determination.

QOS Maintenance Parameter								NPDU-Derived Attributes
b_8	b_7	b_6	b_5	b_4	b_3	b_2	b_1	
1	1	-	-	-	1	0	0	Transit Delay
1	1	-	-	-	1	0	1	Transit Delay
1	1	-	-	-	0	0	0	Expense
1	1	-	-	-	0	1	0	Expense
1	1	-	-	-	0	1	1	Residual Error
1	1	-	-	-	1	1	1	Residual Error

corresponding instance in an FIB's RIB-Att, and, depending on the type of the NPDU-Derived Distinguishing Attribute:

LOCALLY DEFINED QOS: The NSAP Address prefixes and QoS Values are identical.

SECURITY: The same Security Authority is identified in each case.

Provided that such a RIB-Att can be found then the contents is inspected to find an entry such that:

a) the NLRI contains the NPDU's destination NSAP Address, or an NSAP Address prefix which is a prefix of the NPDU's destination NSAP Address;

b) the subnetwork hop's priority, if present, is less than or equal to the NPDU's priority

c) with reference to the applicable Security Policy rules contained in the PIB, the subnetwork hop provides sufficient protection for the NPDU, and the NPDU is permitted to use the subnetwork hop.

d) when a type specific NPDU-Derived Distinguishing Attribute has been selected by a rule specified by a QoS Authority from a source or destination specific QoS Maintenance parameter, then an additional matching rule may also be specified that determines whether the value of the QoS metric is acceptable.

If such a RIB-Att or entry cannot be found, then perform the following procedure in the order indicated, terminating when either a match is found or all three steps have been executed:

a) if the NPDU's security parameter does not express a requirement for protection, the SECURITY attribute may be removed from the NPDU-Derived Distinguishing attributes, and the above procedures repeated in order to find a match.

b) the PRIORITY attribute may be removed from the NPDU-Derived Distinguishing attributes, and the above procedures repeated in order to find a match.

c) LOCALLY DEFINED QOS, EXPENSE, TRANSIT DELAY, or RESIDUAL ERROR (only one of which can be present in a valid set of distinguishing attributes) may be removed from the NPDU-Derived Distinguishing attributes, and the above procedures repeated to find a match.

NOTE 33: If no match was found in the first two steps, the third step will reduce the NPDU-Derived distinguishing attributes to either an empty set or a single security attribute. In the first case, the empty set will match the Empty RIB-Att; in the second case, there can be either a match or a mismatch with the security parameter.

8.4 Forwarding to External Destinations

If the destination address of the incoming NPDU depicts a system located in a different routeing domain from the receiving BIS, then the receiving BIS shall use the FIB identified by the FIB-Att that matches the NDPU-derived Distinguishing Attributes of the incoming NPDU. The incoming NPDU shall be forwarded based on the longest address prefix that matches (as in 7.1.2.2) the destination NSAP address of the incoming NPDU, as follows:

a) If the entry in the inter-domain FIB that corresponds to the destination address of the incoming NPDU contains a NEXT_HOP entry that identifies a BIS which is located on at least one common subnetwork with the local BIS, then the NPDU shall be forwarded directly to the BIS indicated in the NEXT_HOP entry.

b) If the entry in the inter-domain FIB that corresponds to the destination address of the incoming NPDU contains a NEXT_HOP entry that identifies a BIS which is not located on at least one

common subnetwork with the local BIS, then the local BIS has the following options:

1) **Encapsulate the NPDU:** The local BIS may encapsulate the NPDU, using its own NET as the source address and the NET of the next-hop BIS as the destination address. Copy the following, when present in the header of the encapsulated (inner) NPDU, to the header of the encapsulating (outer) NPDU: QOS Maintenance parameter, Segmentation Permitted Flag, Error Report Flag, and PDU Lifetime field. When the inner NPDU is decapsulated, replace its PDU Lifetime field with PDU Lifetime field of the outer NPDU. The encapsulated NPDU shall then be handed over to the intra-domain routeing protocol.

   NOTE 34: It is a local responsibility to insure that the NPDU is encapsulated appropriately for the RD's intra-domain protocol. Since this international standard does not mandate the use of a specific intra-domain protocol, encapsulation details for specific intra-domain protocols are outside its scope.

2) **Use Paths Provided by the Intra-domain Routeing Protocol:** The local BIS may query the intra-domain FIB to ascertain if the intra-domain protocol is aware of a route to the destination system[4]. If there is an intra-domain route that supports the QOS Maintenance parameter of the NPDU and will deliver the NPDU to the appropriate next-hop BIS, then the NPDU may be forwarded along this route.

   NOTE 35: This case makes use of the intra-domain protocol's knowledge of suitable paths through the local RD which support the specified QOS parameter. It does not require encapsulation of the NPDU.

   Details of the mapping between the QOS parameters of used by a given intra-domain protocol and the QOS Maintenance parameter of the NPDU must be determined by the intra-domain routeing protocol; this mapping is not within the scope of IDRP.

9. Interface to ISO 8473

For the purposes of its interface to ISO 8473, a BIS shall be regarded as being comprised of a co-resident ES and IS. The protocol described in this international standard, although still within the Network layer, shall access the service provided by ISO 8473 through an NSAP in the ES part of the BIS. Hence, a BIS's NET shall, for the purposes of communication using ISO 8473, be regarded as NSAP address.

The protocol described in this international standard interfaces at its lower boundary to ISO 8473 using the ISO 8473 primitives shown in Table 5.

The parameters associated with the N-UNITDATA request primitive are:

— *Destination NSAP Address* is the NET of the BIS to which this BISPDU is being sent (that is, the remote BIS)

— *Source NSAP Address* is the NET the BIS that is sending this BISPDU (that is, the local BIS)

— *QOS* is the quality of service parameter for this BIS-BIS connection

— *Userdata* is an ordered sequence of octets which comprise the BISPDU to be sent to the remote BIS

The parameters associated with the N-UNITDATA indication primitive are:

— *Destination NSAP Address* is the NET of the BIS to which this BISPDU is being sent

— *Source NSAP Address* is the NET the BIS that is sending this BISPDU (that is, the remote BIS)

— *QOS* is the quality of service parameter for this BIS-BIS connection

— *Userdata* is an ordered sequence of octets which comprise the BISPDU being delivered to the local BIS

9.1 Use of NLSP over ISO 8473.

EDITOR'S NOTE: The inclusion in this international standard of the material contained in this clause is contingent upon the progression of the Network Layer Security Protocol to DIS status.

The network layer security protocol described in CD 11577 may be used over the ISO 8473 protocol to provide security protection of IDRP exchanges. In this

[4] For example, if ISO 10589 were used as the intra-domain routeing protocol, it would be able to calculate path segments through the RD for systems contained in its "reachable address prefixes".

Table 5. IDRP-CL Primitives	
Primitive	**Parameters**
N-UNITDATA Request	Destination NSAP Address Source NSAP Address QOS Userdata
N-UNITDATA Indication	Destination NSAP Address Source NSAP Address QOS Userdata

case, the NLSP-UNITDATA request and indication service primitives are used instead of the N-UNITDATA service primitives as described in 9 above. The use of BN-UNITDATA notional service primitives by CD 11577 is then mapped onto the use by ISO 8473 of the equivalent N-UNITDATA primitives.

IDRP may control the protection requirements through the use of the NLSP-UNITDATA request and indication primitives or the NLSP QOS Protection parameter; or protection may be imposed by NLSP through the use of security association management.

10. Constants

This constants used by the protocol defined in this international standard are enumerated in Table 6.

Table 6. Architectural Constants of IDRP		
Name of Constant	**Value**	**Description**
Inter-domain Routeing Protocol Identifier	X'85'	The SPI for the protocol described in this international standard
MinBISPDULength	30	The size in octets of the smallest allowable BISPDU.
MinRDOriginationInterval	15 minutes	The minimum time between successive UPDATE PDUs advertising routeing information about the local RD
Jitter	0.25	The factor used to compute jitter according to clause 7.17.3.3.
MaxCPUOverloadPeriod	1 hour	Maximum time in which a BIS can remain CPU-overloaded before terminating its BIS-BIS connections.
CloseWaitDelay	150 seconds	The time that a FSM remains in CLOSE-WAIT state before entering the CLOSED state.

11. System Management and GDMO Definitions

The operation of the inter-domain routeing functions in a BIS may be monitored and controlled using System Management. This clause contains management specification for IDRP, expressed in the GDMO notation defined in ISO 10165-4.

11.1 Name Binding

idrpConfig-networkEntity NAME BINDING

SUBORDINATE OBJECT CLASS idrpConfig AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS "ISO/IEC 10733 : 19xx": networkEntity;
WITH ATTRIBUTE "ISO/IEC xxxx": idrpConfigId;
CREATE WITH-AUTOMATIC-INSTANCE-NAMING;
DELETE ONLY-IF-NO-CONTAINED-OBJECTS;
REGISTERED AS { IDRP.nboi idrpConfig-networkEntity(1) };

adjacentBIS-idrpConfig NAME BINDING

SUBORDINATE OBJECT CLASS adjacentBIS AND SUBCLASSES;
NAMED BY SUPERIOR OBJECT CLASS idrpConfig AND SUBCLASSES;
WITH ATTRIBUTE bisNet;
BEHAVIOUR iDRPConfig-adjacentBIS-idrpConfig-B BEHAVIOUR

DEFINED AS This name binding attribute identifies a BIS to BIS connection information block. One of these blocks of data should exist per remote BIS that this local BIS exchanges BISPDUs with.;;

REGISTERED AS { IDRP.nboi adjacentBIS-idrpConfig(2) };

11.2 Local BIS Managed Objects for IDRP

idrpConfig MANAGED OBJECT CLASS

DERIVED FROM "Rec. X.721 | ISO/IEC 10165-2 : 1992": top;
CHARACTERIZED BY idrpConfigPkg PACKAGE
BEHAVIOUR iDRPBasicImportedAlarmNotifications-B

BEHAVIOUR DEFINED AS Imports the communicationsAlarm notification from ISO/IEC 10165-2. It is used to report the following protocol events:

errorBISPDUsent: Generated when a BISPDU is received with an error in its format. The significance sub-parameter of each item of additionalInformation shall be set to the value False' (i.e., not significant) so that a managing system receiving the event report will be less likely to reject it. The probableCause shall be set to NLM.communicationsProtocolError. The perceivedSeverity shall be set to 'Minor'. A subsequent communicationsAlarm with a perceivedSeverity value of 'Cleared' shall not be generated. No other fields or parameters shall be used, with the exception of further parameters in the AdditionalInformation field, as follows:

a) RemoteBIS-NET for BIS-BIS connection--using the notificationRemoteBIS-NET parameter
b) BISPDU error code (see 6.4 and 7.20)--this reports the error code that will be sent in the ERROR PDU using the parameter "notificationBISPDUerrorcode".
c) BIS error subcode (see 6.4 and 7.20)--this reports the subcode that will be sent using the parameter "notificationBISerrorsubcode".
d) BISPDU error information (see 6.4 and 7.20)--this reports the data from the received BISPDU that will be used to diagnose the problem for the Notification. The parameter "notificationBISpduerrorinfo" will be used to report this information.

OpenBISpduRDCError: generated when an OPEN BISPDU is received from another BIS in the same routeing domain, and the remote BIS is not a member of identically the same confederations as the local BIS. The significance sub-parameter of each item of additionalInformation shall be set to the value 'False' (i.e., not significant) so that a managing system receiving the event report will be less likely to reject it. The probableCause shall be set to NLM.communicationsProtocolError. The perceivedSeverity shall be set to 'Minor'. A subsequent communicationsAlarm with a

perceivedSeverity value of 'Cleared' shall not be generated. No other fields or parameters shall be used, with the exception of further parameters in the AdditionalInformation field, as follows:

a) Remote BIS NET for this BIS-BIS connection--using the "notificationRemoteBIS-NET" parameter.
b) Remote BIS Routeing Domain Confederation (RDC) information using the "notificationRemoteRDCconfig" parameter.
c) Local BIS Routeing Domain Confederation (RDC) information using the "notificationLocalRDCconfig" parameter.

errorBISPDUconnectionclose: generated when an ERROR PDU has been received from a remote BIS. The significance sub-parameter of each item of additionalInformation shall be set to the value 'False' (i.e., not significant) so that a managing system receiving the event report will be less likely to reject it. The probableCause shall be set to NLM.communicationsProtocolError. The perceivedSeverity shall be set to 'Minor'. A subsequent communicationsAlarm with a perceivedSeverity value of 'Cleared' shall not be generated. No other fields or parameters shall be used, with the exception of further parameters in the AdditionalInformation field, as follows:

a) RemoteBIS-NET for BIS-BIS connection--using the "notificationRemoteBIS-NET" parameter
b) BISPDU error code (see 6.4 and 7.20)--this reports the error code that will be sent in the ERROR PDU using the parameter "notificationBISpduerrorcode".
c) BIS error subcode (see 6.4 and 7.20)--this reports the subcode that will be sent using the parameter "notificationBISerrorsubcode".
d) BISPDU error information (see 6.4 and 7.20)--this reports the data from the received BISPDU that will be used to diagnose the problem for the Notification. The parameter "notificationBISPDerrorinfo" will be used to report this information.

CorruptAdjRIBIn: generated when the local method of checking the Adj-RIB-In has found an error. All Adj-RIBs-In are being purged. The significance sub-parameter of each item of additionalInformation shall be set to the value 'False' (i.e., not significant) so that a managing system receiving the event report will be less likely to reject it. The probableCause shall be set to NLM.communicationsProtocolError. The perceivedSeverity shall be set to 'Minor'. A subsequent communicationsAlarm with a perceivedSeverity value of 'Cleared' shall not be generated. No other fields or parameters shall be used, with the exception of further parameters in the AdditionalInformation field, as follows:

a) MaxRIBIntegrityCheck attribute for this BIS
b) The remote BIS associated with this Adjacent RIB in the parameter "notificationRemoteBIS-NET".

PacketBomb: generated when the local BIS received a BISPDU other than an OPEN PDU, from an unknown BIS. The Source-BIS-NET is reported in the AdditionalInformation field using the "notificationSourceBIS-NET parameter. The significance sub-parameter of each item of additionalInformation shall be set to the value 'False' (i.e., not significant) so that a managing system receiving the event report will be less likely to reject it. The probableCause shall be set to NLM.communicationsProtocolError. The perceivedSeverity shall be set to 'Minor'. A subsequent communicationsAlarm with a perceivedSeverity value of 'Cleared' shall not be generated. No other fields or parameters shall be used, with the exception of further parameters in the AdditionalInformation field, as follows. These parameters are created from the OPEN PDU values:

a) notificationSourceBISNET--NET of remote BIS sending packet bomb
b) notificationSourceBISrdi--RDI of remote BIS sending packet bomb
c) notificationSourceBISrdc--RDC information for remote BIS sending packet bomb

connectRequestBISUnknown: generated when the local BIS has received an OPEN BISPDU from an unknown BIS. The significance sub-parameter of each item of additionalInformation shall be set to the value 'False' (i.e., not significant) so that a managing system receiving the event report will be less likely to reject it. The probableCause shall be set to NLM.communicationsProtocolError. The perceivedSeverity shall be set to 'Minor'. A subsequent communicationsAlarm with a perceivedSeverity value of 'Cleared' shall not be generated. No other fields or parameters shall be used, with the exception of further parameters in the AdditionalInformation field, as follows. These parameters are created from the OPEN PDU values:

a) notificationSourceBISNET--NET of remote BIS sending packet bomb
b) notificationSourceBISrdi--RDI of remote BIS sending packet bomb
c) notificationSourceBISrdc--RDC information for remote BIS sending packet bomb

EnterFSMStateMachine: generated when the IDRP FSM state machine used to communicate with another BIS is started. The RemoteBis-NET is reported in the additionalInformation field using the ″notificationRemoteBIS-NET″ parameter. The significance sub-parameter of each item of additionalInformation shall be set to the value 'False' (i.e., not significant) so that a managing system receiving the event report will be less likely to reject it. The probableCause shall be set to NLM.communicationsProtocolError. The perceivedSeverity shall be set to 'Minor'. A subsequent communicationsAlarm with a perceivedSeverity value of 'Cleared' shall not be generated. No other fields or parameters shall be used, with the exception of further parameters in the AdditionalInformation field, as follows:

iDRPBasicImportedInfoNotifications-B

BEHAVIOUR DEFINED AS Imports the communicationsInformation notification from ISO/IES 10165-2. It is used to report the following protocol events:

EnterFSMState: generated when a BIS starts the IDRP FSM state machine to establish a connection with a remote BIS. The RemoteBis-NET is reported in the AdditionalInformation field using the ″notificationRemoteBIS-NET″ parameter. The significant subparameter of each item of AdditionalInformation shall be set to ″false″ (that is, not significant) so that a managing system receiving the event report will be less likely to reject it.

FSMStateChange: generated when the IDRP FSM used to communicate with another BIS transitions from one state to another. The RemoteBis-NET is reported in the AdditionalInformation field using the ″notificationRemoteBIS-NET″ parameter. The significant sub-parameter of each item ofAdditionalInformation shall be set to ″false″ (that is, not significant) so that a managing system receiving the event report will be less likely to reject it.;;

ATTRIBUTES

authenticationTypeCode GET,
capacity GET,
closeWaitDelayTimer GET
externalBISNeighbor GET-REPLACE,
holdTime GET,
holdTimer GET,
internalBIS GET-REPLACE,
internalSystems GET-REPLACE,
intraIS GET-REPLACE,
keepAliveTimer GET,
localRDI GET-REPLACE,
localSNPA GET-REPLACE,
locExpense GET,
maxCPUOverloadTimer GET
maxPDULocal GET,
maxRIBIntegrityCheck GET;
maxRIBIntegrityTimer GET
minRDOriginationTimer GET
minRouteAdvertisementTimer GET
multiExit GET-REPLACE,
priority GET,
rdcConfig GET-REPLACE,
rdLRE GET,
rdTransitDelay GET,
retransmissionTime GET,
ribAttsSet GET,
routeServer GET-REPLACE,
version GET,

ACTIONS

″GMI″:activate
″GMI″:deactivate

NOTIFICATIONS ″REC X.721 | ISO/IEC 10165-2:1992″: communicationsAlarm

notificationBISerrorssubcode
notificationBISpduerrorcode
notificationBISpduerrorinfo

```
        notificationLocalRDCconfig
        notificationMaxAdjRibIntegritycheck
        notificationRemoteBIS-NET
        notificationRemoteRDCconfig
        notificationSourceBISNET
        notificationSourceBISrdc
        notificationSourceBISrdi,
    "REC X.723 | ISO/IEC 10165-5: 1992": communicationsInformation notificationRemoteBIS-NET;;;
    REGISTERED AS {IDRP.moi iDRPConfig (1)};
```

11.3 Adjacent BIS Peer Managed Objects

```
adjacentBIS MANAGED OBJECT CLASS
    DERIVED FROM "Rec. X.721 | ISO/IEC 10165-2 : 1992": top;
    CHARACTERIZED BY adjacentBISPkg PACKAGE
    ATTRIBUTES
        bisNegotiatedVersion GET,
        bisNet GET,
        bisPeerSNPAs GET,
        bisRDC GET,
         DEFAULT VALUE NULL,
        bisRDI GET,
         DEFAULT VALUE NULL,
        keepAlivesSinceLastUpdate GET;
        lastAckRecv GET
         INITIAL VALUE 0,
        lastAckSent GET
         INITIAL VALUE 0,
        lastSeqNoRecv GET
         INITIAL VALUE 0,
        lastSeqNoSent GET,
         INITIAL VALUE 0
        listenForOPEN GET,
        maxPDUPeer GET,
        outstandingPDUs
        state GET,
        totalBISPDUsIn GET,
        totalBISPDUsOut GET,
        updatesIn GET,
        updatesOut GET,
    ATTRIBUTE GROUPS
        "GMI":counters
            updatesIn
            updatesOut
            totalBISPDUsIn
            totalBISPDUsOut
            keepAlivesSinceLastUpdate,
        "DMI":state
            state
    ACTIONS
        "GMI":activate
        "GMI":deactivate
    REGISTERED AS { IDRP.moi adjacentBIS(2) };
```

11.4 Attribute Definitions

authenticationTypeCode ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.AuthenticationCode;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR authenticationCode-B
BEHAVIOUR DEFINED AS Indication of which authentication mechanism will be used;;
REGISTERED AS { IDRP.atoi authenticationCode(1) };

bisNegotiatedVersion ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.BisNegotiatedVersion;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR bisNegotiatedVersion-B
BEHAVIOUR DEFINED AS The negotiated version of IDRP protocol this BIS to BIS connection is using.;;
REGISTERED AS { IDRP.atoi bisNegotiatedVersion(2) };

bisNet ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.BisNet;
MATCHES FOR EQUALITY;
BEHAVIOUR bisNet-B
BEHAVIOUR DEFINED AS The NET of the remote BIS of this BIS to BIS connection.;;
REGISTERED AS { IDRP.atoi bisNet(3) };

bisPeerSNPAs ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.BisPeersSNPAs;
MATCHES FOR EQUALITY;
BEHAVIOUR bisPeerSNPAs-B
BEHAVIOUR DEFINED AS The SNPAs announced by the remote BIS of this BIS to BIS connection.;;
REGISTERED AS { IDRP.atoi bisPeerSNPAs(4) };

bisRDC ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.Rdcgroup;
MATCHES FOR EQUALITY;
BEHAVIOUR bisRDC-B
BEHAVIOUR DEFINED AS ″The RDC the remote BIS belongs to in this BIS to BIS connection.″;;
REGISTERED AS { IDRP.atoi bisRDC(5) };

bisRDI ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.Rdi;
MATCHES FOR EQUALITY;
BEHAVIOUR bisRDI-B
BEHAVIOUR DEFINED AS The RDI of the remote BIS of this BIS to BIS connection.;;
REGISTERED AS { IDRP.atoi bisRDI(6) };

capacity ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.Capacity;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR capacity-B
BEHAVIOUR DEFINED AS The traffic carrying capacity of this Routeing Domain.;;
REGISTERED AS { IDRP.atoi capacity(7) };

closeWaitDelayTimer ATTRIBUTE

DERIVED FROM ″GMI″:timer;
BEHAVIOUR closeWaitDelayTimer-B
BEHAVIOUR DEFINED AS The timer that measures in seconds the time that has elapsed since the BIS FSM entered the CLOSE-WAIT state. Upon timer expiration, the BIS FSM will enter the CLOSED state;;

REGISTERED AS { IDRP.atoi closeWaitDelayTimer(8) };

externalBISNeighbor ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.BISGroup;
MATCHES FOR EQUALITY;
BEHAVIOUR externalBISNeighbor-B
BEHAVIOUR DEFINED AS The set of NETs which identify the BISs in adjacent routeing domain that are reachable via a single subnetwork hop.;;
REGISTERED AS { IDRP.atoi externalBISNeighbor(9) };

holdTime ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.Holdtime;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR holdTime-B

BEHAVIOUR DEFINED AS The maximum number of seconds that may elapse between the receipt of two successive BISPDUs of any of the following types: KEEPALIVE, UPDATE, RIB CHECKSUM PDUs or RIB REFRESH PDUs;;

REGISTERED AS { IDRP.atoi holdTime(10) };

holdTimer ATTRIBUTE

DERIVED FROM "GMI":timer;
BEHAVIOUR keepAliveTimer-B
BEHAVIOUR DEFINED AS The timer that measures in seconds the time that has elapsed since the reception of the most recent BISPDU of any of the following types: KEEPALIVE, UPDATE, RIB REFRESH;;
REGISTERED AS { IDRP.atoi holdTimer(11) };

internalBIS ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.BISGroup;
MATCHES FOR EQUALITY;
BEHAVIOUR internalBIS-B
BEHAVIOUR DEFINED AS The set of NETs which identify the BISs in this routeing domain;;
REGISTERED AS { IDRP.atoi internalBIS(12) };

internalSystems ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.SystemIdGroup;
MATCHES FOR EQUALITY;
BEHAVIOUR internalSystems-B
BEHAVIOUR DEFINED AS The set of NET and NSAP prefixes that identify the systems in this routeing domain for which the BIS constructs this routeing domain from which the BIS constructs network layer reachability information;;
REGISTERED AS { IDRP.atoi internalSystems(13) };

intraIS ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.BISGroup;
MATCHES FOR EQUALITY;
BEHAVIOUR intraIS-B
BEHAVIOUR DEFINED AS The set of NETs of the ISs to which the local BIS may deliver an inbound NPDU whose destination lies within the BIS's routeing domain. These ISs must be located on the same common subnetwork as this local BIS, and must be capable of delivering NPDUs to destinations that are located within the local BIS's routeing domain.;;
REGISTERED AS { IDRP.atoi intraIS(14) };

keepAlivesSinceLastUpdate ATTRIBUTE

DERIVED FROM "GMI":nonWrapping64BItCounter;
BEHAVIOUR keepAlivesSinceLastUpdate-B

```
        BEHAVIOUR DEFINED AS The number of KEEPALIVE BISPDUS received by this BIS from the remote BIS
        since this last UPDATE BISPDU.;;
    REGISTERED AS { IDRP.atoi keepAlivesSinceLastUpdate(15) };

keepAliveTimer ATTRIBUTE

    DERIVED FROM "GMI":timer;
    BEHAVIOUR keepAliveTimer-B
        BEHAVIOUR DEFINED AS The timer that measures in seconds the time that has elapsed since the pre-
        vious KEEPALIVE PDU was received by the local BIS.  Upon its expiration, the BIS will send a BISPDU to
        its peer BIS;;
    REGISTERED AS { IDRP.atoi keepAliveTimer(16) };

lastAckRecv ATTRIBUTE

    WITH ATTRIBUTE SYNTAX INTEGER;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR lastAckRecv-B
        BEHAVIOUR DEFINED AS The number of the last ack received from the remote BIS by this local BIS on
        this BIS to BIS connection.;;
    REGISTERED AS { IDRP.atoi lastAckRecv(17) };

lastAckSent ATTRIBUTE

    WITH ATTRIBUTE SYNTAX INTEGER;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR lastAckSent-B
        BEHAVIOUR DEFINED AS The number of the last ack sent to the remote BIS from this local BIS on this
        BIS to BIS connection.;;
    REGISTERED AS { IDRP.atoi lastAckSent(18) };

lastSeqNoRecv ATTRIBUTE

    WITH ATTRIBUTE SYNTAX INTEGER;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR lastSeqNoRecv-B
        BEHAVIOUR DEFINED AS The last sequence number received from the remote BIS by this local BIS on
        this BIS to BIS connection.;;
    REGISTERED AS { IDRP.atoi lastSeqNoRecv(19) };

lastSeqNoSent ATTRIBUTE

    WITH ATTRIBUTE SYNTAX INTEGER;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR lastSeqNoSent-B
        BEHAVIOUR DEFINED AS The last sequence number sent to the remote BIS from this local BIS on this
        BIS to BIS connection.;;
    REGISTERED AS { IDRP.atoi lastSeqNoSent(20) };

listenForOPEN ATTRIBUTE

    WITH ATTRIBUTE SYNTAX BOOLEAN;
    MATCHES FOR EQUALITY;
    BEHAVIOUR listenForOPEN-B
        BEHAVIOUR DEFINED AS The boolean value determines how the Finite State machnine will react to
        reception of an OPEN PDU while it is in the CLOSED state;;
    REGISTERED AS { IDRP.atoi listenForOPEN(21) };

localRDI ATTRIBUTE

    WITH ATTRIBUTE SYNTAX IDRP.Rdi;
    MATCHES FOR EQUALITY;
    BEHAVIOUR localRDI-B
```

```
        BEHAVIOUR DEFINED AS The Routing Domain Identifier for the routeing domain where this BIS is
        located;;
    REGISTERED AS { IDRP.atoi localRDI(22) };

localSNPA ATTRIBUTE

    WITH ATTRIBUTE SYNTAX IDRP.LocalSNPAs;
    MATCHES FOR EQUALITY;
    BEHAVIOUR localSNPA-B
        BEHAVIOUR DEFINED AS The list of SNPAs of this BIS;;
    REGISTERED AS { IDRP.atoi localSNPA(23) };

locExpense ATTRIBUTE

    WITH ATTRIBUTE SYNTAX IDRP.LocExpense;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR locExpense-B
        BEHAVIOUR DEFINED AS The monetary expense of transiting this Routeing Domain. The attribute con-
        tains an indication of cost and the units in which it is calculated;;
    REGISTERED AS { IDRP.atoi locExpense(24) };

maxCPUOverloadTimer ATTRIBUTE

    DERIVED FROM "GMI":timer;
    BEHAVIOUR maxCPUOverloadTimer-B
        BEHAVIOUR DEFINED AS The timer that measures in seconds the time that has elapsed since the local
        BIS has detected that its CPU has become overloaded. See Annex H;;
    REGISTERED AS { IDRP.atoi maxCPUOverloadTimer(25) };

maxPDULocal ATTRIBUTE

    WITH ATTRIBUTE SYNTAX IDRP.MaximumPDUSize;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR maximumPDUsize-B

        BEHAVIOUR DEFINED AS The maximum number of octets that a peer BIS can include in a BISPDU that it
        sends to this BIS. The local BIS informs its peer of this value via the OPEN PDU sent to the peer;;

    REGISTERED AS { IDRP.atoi maximumPDUsize(26) };

maxPDUPeer ATTRIBUTE

    WITH ATTRIBUTE SYNTAX IDRP.MaximumPDUSize;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR maximumPDUsize-B

        BEHAVIOUR DEFINED AS The maximum number of octets that this BIS will include in a BISPDU that it
        sends to its peer BIS. This value is obtained from information in the header of the peer BIS's OPEN
        PDU;;

    REGISTERED AS { IDRP.atoi maximumPDUsize(27) };

maxRIBIntegrityCheck ATTRIBUTE

    WITH ATTRIBUTE SYNTAX IDRP.MaxRIBIntegrityCheck;
    MATCHES FOR EQUALITY, ORDERING;
    BEHAVIOUR maxRIBIntegrityCheck-B
        BEHAVIOUR DEFINED AS The maximum time in seconds between checking of the Adj-RIBs-In by a local
        mechanism. If corrupt Adj-RIB-In is detected, the BIS shall purge the offending Adj-RIB-In;;
    REGISTERED AS { IDRP.atoi maxRIBIntegrityCheck(28) };

maxRIBIntegrityTimer ATTRIBUTE

    DERIVED FROM "GMI":timer;
    BEHAVIOUR maxRIBIntegrityTimer-B
```

BEHAVIOUR DEFINED AS The timer that measures in seconds the time remaining until the Adj-RIBs-In must be checked by a local mechanism. If a corrupt Adj-RIB-In is detected, the BIS shall purge the offending Adj-RIB-In;;
REGISTERED AS { IDRP.atoi maxRIBIntegrityTimer(29) };

minRDOriginationTimer ATTRIBUTE

DERIVVED FROM "GMI":timer;
BEHAVIOUR minRDOriginationTimer-B
BEHAVIOUR DEFINED AS The timer that measures in seconds the time that has elapsed since the advertisement by the local BIS of an UPDATE PDU that reported changes within the local BIS's routeing domain. See clause 7.17.3.2;;
REGISTERED AS { IDRP.atoi minRDOriginationTimer(30) };

minRouteAdvertisementTimer ATTRIBUTE

DERIVED FROM "GMI":timer;
BEHAVIOUR minRouteSelectionTimer-B
BEHAVIOUR DEFINED AS The timer that measures in seconds the time that has elapsed since the advertisement by the local BIS of a better route that was received from a BIS located in another routeing domain. See clause 7.17.3.2. The minimum value is 5 seconds, and the maximum value is 1800 seconds;;
REGISTERED AS { IDRP.atoi minRouteAdvertisementTimer(31) };

multiExit ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.Boolean;
MATCHES FOR EQUALITY;
BEHAVIOUR multiExit-B
BEHAVIOUR DEFINED AS The indication whether this BIS will use the MULTI_EXIT_DISC attribute to decide between otherwise identical routes. The multiExit parameter is used as the default value for the "multi_exit_disc" function in policy decisions.;;
REGISTERED AS { IDRP.atoi multiExit(32) };

outstandingPDUs ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.OutstandingPdus;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR outstandingPDUs-B
BEHAVIOUR DEFINED AS The maximum number of BISPDUs that may be sent to this BIS without receiving an acknowledgement;;
REGISTERED AS { IDRP.atoi outstandingPDUs(33) };

priority ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.Priority;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR priority-B
BEHAVIOUR DEFINED AS The lowest value of ISO 8473 priority parameter that this RD will provide forwarding services for;;
REGISTERED AS { IDRP.atoi priority(34) };

rdcConfig ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.Rdcgroup;
MATCHES FOR EQUALITY;
BEHAVIOUR rdcConfig-B
BEHAVIOUR DEFINED AS All of the Routing Confederations to which the RD of this BIS belongs and the nesting relationships that are in force between them. The nesting relationships are described as a sequence of sets of RDC Identifiers;;
REGISTERED AS { IDRP.atoi rdcConfig(35) };

rdLRE ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.Rdlre;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR rdLRE-B
BEHAVIOUR DEFINED AS A quantity that is proportional to the average error rate of a Routeing Domain and is expressed as an an integer in the range from 0 to $2^{32} - 1$. The actual error rate is equal to the integer value divided by $2^{32} - 1$.;;
REGISTERED AS { IDRP.atoi rdLRE(36) };

rdTransitDelay ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.RDTransitDelay;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR rdTransitDelay-B
BEHAVIOUR DEFINED AS The estimated average delay across a Routeing Domain in units of 2 ms.;;
REGISTERED AS { IDRP.atoi rdTransitDelay(37) };

retransmissionTime ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.RetransmissionTime;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR retransmissionTime-B
BEHAVIOUR DEFINED AS The Number of seconds of between KEEPALIVE messages if no other traffic is sent;;
REGISTERED AS { IDRP.atoi retransmissionTime(38) };

ribAttsSet ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.RibAttsSet;
MATCHES FOR EQUALITY;
BEHAVIOUR ribAttsSet-B
BEHAVIOUR DEFINED AS The set of Rib Attributes supported by this BIS.;;
REGISTERED AS { IDRP.atoi ribAttsSet(39) };

routeServer ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.Boolean;
MATCHES FOR EQUALITY;
BEHAVIOUR routeServer-B
BEHAVIOUR DEFINED AS The indication whether this BIS may set the ″IDRP_Server_Allowed″ field in the NEXT_HOP attribute to X″FF″ for BIS to BIS UPDATE BISPDUs. If this variable is true then in accordance with local policy, the IDRP_Server_Allowed field may be set on some UPDATE BISPDUs that this BIS sends. If this attribute is set to false, then no UPDATE BISPDUs will be sent by this BIS with NEXT_HOP attributes containing an ″IDRP_Server flag″ equal to X″FF″.;;
REGISTERED AS { IDRP.atoi routeServer(40) };

state ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.State;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR state-B
BEHAVIOUR DEFINED AS The current state of BIS to BIS communication in the local BIS.;;
REGISTERED AS { IDRP.atoi state(41) };

totalBISPDUsIn ATTRIBUTE

DERIVED FROM ″GMI″:nonWrapping64BItCounter;
BEHAVIOUR totalBISPDUsIn-B
BEHAVIOUR DEFINED AS The number of BISPDUS received by this BIS from the remote BIS on this BIS to BIS connection.;;
REGISTERED AS { IDRP.atoi totalBISPDUsIn(42) };

totalBISPDUsOut ATTRIBUTE

DERIVED FROM ″GMI″:nonWrapping64BItCounter;

BEHAVIOUR totalBISPDUsOut-B
BEHAVIOUR DEFINED AS The number of BISPDUS received by this BIS from the remote BIS on this BIS to BIS connection.;;
REGISTERED AS { IDRP.atoi totalBISPDUsOut(43) };

updatesIn ATTRIBUTE

DERIVED FROM "GMI":nonWrapping64BitCounter;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR updatesIn-B
BEHAVIOUR DEFINED AS The number of UPDATE BISPDUs received by this BIS on this BIS to BIS connection.;;
REGISTERED AS { IDRP.atoi updatesIn(44) };

updatesOut ATTRIBUTE

DERIVED FROM "GMI":nonWrapping64BItCounter;
BEHAVIOUR updatesOut-B
BEHAVIOUR DEFINED AS The number of UPDATE BISPDUs sent by this BIS on this BIS to BIS connection.;;
REGISTERED AS { IDRP.atoi updatesOut(45) };

version ATTRIBUTE

WITH ATTRIBUTE SYNTAX IDRP.Version;
MATCHES FOR EQUALITY, ORDERING;
BEHAVIOUR version-B
BEHAVIOUR DEFINED AS The version of IDRP protocol this machine defaults to using.;;
REGISTERED AS { IDRP.atoi version(46) };

11.5 Parameter Definitions

notificationRemoteBIS-NET PARAMETER

CONTEXT EVENT-INFO;
WITH SYNTAX IDRP.RemoteBISNetActionReply;
BEHAVIOUR notificationRemoteBIS-NET-B
BEHAVIOUR DEFINED AS The NET of the Remote BIS that this local BIS is starting IDRP protocol communication with.;;
REGISTERED AS { IDRP.poi notificationRemoteBIS-NET(1) };

notificationBISpduerrorcode PARAMETER

CONTEXT EVENT-INFO;
WITH SYNTAX IDRP.notificationBISpduerrorCode;
BEHAVIOUR notificationBISpduerrorcode-B
BEHAVIOUR DEFINED AS The error code indicating what type of error occurred in the BIS PDU.;;
REGISTERED AS { IDRP.poi notificationBISpduerrorcode(2) };

notificationBISerrorsubcode PARAMETER

CONTEXT EVENT-INFO;
WITH SYNTAX IDRP.BispduErrorSubcode;
BEHAVIOUR notificationBISerrorsubcode-B
BEHAVIOUR DEFINED AS The error code indicating what type of error within the major error type occurred in the BIS PDU.;;
REGISTERED AS { IDRP.poi notificationBISerrorsubcode(3) };

notificationBISPDUerrorinfo PARAMETER

CONTEXT EVENT-INFO;
WITH SYNTAX IDRP.BispduErrorInfo;
BEHAVIOUR notificationBISPDUerrorinfo-B

```
        BEHAVIOUR DEFINED AS The additional information from original PDU that indicated an error in the BIS
        PDU.;;
    REGISTERED AS { IDRP.poi notificationBISPDUerrorinfo(4) };

notificationRemoteRDCConfig PARAMETER

    CONTEXT EVENT-INFO;
    WITH SYNTAX  IDRP.RemoteRDCConfig;
    BEHAVIOUR notificationRemoteRDCConfig-B
        BEHAVIOUR DEFINED AS The Routing Domain Confederation (RDC) informa  tion from the remote BIS on
        this BIS to BIS communication.;;
    REGISTERED AS { IDRP.poi notificationRemoteRDCConfig(5) };

notificationLocalRDCConfig PARAMETER

    CONTEXT EVENT-INFO;
    WITH SYNTAX  IDRP.LocalRDCConfig;
    BEHAVIOUR notificationLocalRDCConfig-B
        BEHAVIOUR DEFINED AS The Routing Domain Confederation (RDC) informa  tion from this local BIS on
        this BIS to BIS communcation.;;
    REGISTERED AS { IDRP.poi notificationLocalRDCConfig(6) };

maxRIBIntegrityCheck PARAMETER

    CONTEXT EVENT-INFO;
    WITH SYNTAX  IDRP.MaxRIBIntegrityCheck;
    BEHAVIOUR maxRIBIntegrityCheck-B
        BEHAVIOUR DEFINED AS The maximum number of integrity checks detected before reporting the event
        to systems management;;
    REGISTERED AS { IDRP.poi maxRIBIntegrityCheck(7) };

notificationSourceBISNET PARAMETER

    CONTEXT EVENT-INFO;
    WITH SYNTAX  IDRP.NetworkEntityTitle;
    BEHAVIOUR notificationSourceBISNET-B
        BEHAVIOUR DEFINED AS NET of the remote BIS that sent a packet bomb to the local BIS;;
    REGISTERED AS { IDRP.poi notificationRemoteBISNET(8) };

notificationSourceBISrdi PARAMETER

    CONTEXT EVENT-INFO;
    WITH SYNTAX  IDRP.Rdi;
    BEHAVIOUR notificationSourceBISRdi-B
        BEHAVIOUR DEFINED AS RDI of the remote BIS that sent a packet bomb to the local BIS;;
    REGISTERED AS { IDRP.poi notificationRemoteBISrdi(9) };

notificationSourceBISrdc PARAMETER

    CONTEXT EVENT-INFO;
    WITH SYNTAX  IDRP.Rdi;
    BEHAVIOUR notificationSourceBISrdc-B
        BEHAVIOUR DEFINED AS RDC of the remote BIS that sent a packet bomb to the local BIS;;
    REGISTERED AS { IDRP.poi notificationRemoteBISrdc(10) };
```

11.6 ASN.1 Modules

```
IDRP{joint-iso-ccitt network-layer(13) management(0) iDRP(3)
    asn1Module(2) 0}
DEFINITIONS::=BEGIN
  -- object identifier definitions
idrpoi OBJECT IDENTIFIER ::= {NLM.nl iDRP(3)}
sseoi OBJECT IDENTIFIER ::={idrpoi standSpecificExtensions(0)}
```

```
moi OBJECT IDENTIFIER ::= {idrpoi objectClass (3)}
poi OBJECT IDENTIFIER ::= {idrpoi package (4)}
proi OBJECT IDENTIFIER ::={idrpoi parameter(5)}
nboi OBJECT IDENTIFIER ::={idrpoi nameBinding (6)}
atoi OBJECT IDENTIFIER ::={idrpoi attribute (7)}
agoi OBJECT IDENTIFIER ::={idrpoi attributeGroup (8)}
acoi OBJECT IDENTIFIER ::={idrpoi action (9)}
noi OBJECT IDENTIFIER ::={idrpoi notification (10)}

--
--object identifiers for notification parameters
--

se OBJECT IDENTIFIER ::= {sseoi specificProblems(3)}

errorBISPDUsent OBJECT IDENTIFIER ::= {se errorBISPDU0(0)}
openBISpduRDCerror OBJECT IDENTIFIER ::= {se errorBISPDU1(1)}
errorBISPDUconnectionclose OBJECT IDENTIFIER ::= {se
    errorBISPDU2(2)}
corruptAdjRIBIn OBJECT IDENTIFIER ::= {se errorBISPDU3(3)}
packetBomb OBJECT IDENTIFIER ::= {se errorBISPDU4(4)}
enterFSMstate OBJECT IDENTIFIER ::= {se errorBISPDU5(5)}
fSMStateChange OBJECT IDENTIFIER ::= {se errorBISPDU6(6)}

--
--ASN1 Types and Values
--

AuthenticationCode ::=ENUMERATED{
  integrityOnly(0),
  integrityPlusAuthentication(1)
  integrityPlusSecretText(2)}
Authtype ::=AuthenticationCode
BISGroup ::= SET OF NetworkEntityTitle
BisNet ::= NetworkEntityTitle
BisNegotiatedVersion ::=Version
BispduErrorCode::= ENUMERATED {
  oPENPDUerror (1),
  uPDATEPDUError (2),
  holdtimerExpired (3)}
BispduErrorSubcode ::= CHOICE {
  operr [] IMPLICIT Openerrorsubcode,
  uperr [1] IMPLICIT Updateerrorsubcode}

BispduErrorInfo ::=OCTET STRING(SIZE(1..50))
 --50 bytes of original message are saved
BisPeersSNPAs ::= SNPAaddresses
Boolean ::= BOOLEAN
Capacity ::=INTEGER(1..255)
EndSystemNSAP ::= OCTET STRING(SIZE(1..20))
ESPrefix ::= NSAPprefix
Expensevalue ::=Locexpense
GLOBAL ::= ENUMERATED ( delay(0), expense(1), capacity(3), error(4) }
Holdtime ::=INTEGER(1..65535)
KeepaliveSincelastupdupdate ::=INTEGER(1..4294967295)
Lastseqnosent ::=INTEGER(1..4294967295)
Lastseqnorecv ::=INTEGER(1..4294967295)
Lastacksent ::=INTEGER(1..4294967295)
Lastackrecv ::=INTEGER(1..4294967295)
LocExpense ::= INTEGER(1..65535)
```

```
LocalRDCConfig ::=Rdcgroup
LocalSNPAs ::= SNPAaddresses
MaxPDULocal ::=INTEGER(1..65535)
MaxPDUPeer ::=INTEGER(1..65535)
MaxRIBIntegrityCheck ::=INTEGER(1..65535)
Metriclength ::=INTEGER(1..255)
Metricvalue ::=OCTET STRING(SIZE(1..255))
NSAPprefixLength ::=INTEGER(1..160)
NSAPprefix ::= BIT STRING(SIZE(1..160))
NetworkEntityTitle ::=OCTET STRING(SIZE(1..20))
NETPrefix ::= NSAPprefix
  --0..(2 exp 64)-1
NotificationInfo ::=SET OF Parameter
Openerrorsubcode ::=ENUMERATED {
  unsupportedVersionnumber (1),
  badMaxPDUsize (2),
  badOutstandingPDUs (3),
  badPeerRD (4),
  unsupportedAuthenticationcode (5),
  authenticationFailure (6),
  badRIB-AttrsSet (7),
  rDCmismatch (8)}
OutstandingPdus ::=INTEGER(0..255)
Priority ::= INTEGER(0..14)
QOS ::= CHOICE { global[0] EXPLICIT GLOBAL,
            ssQOS[1] EXPLICIT QOSTV,
            dsQOS[2] EXPLICIT QOSTV }
QOSlength ::= INTEGER(1..255)
QOSTV ::= SEQUENCE { preflgth NSAPorefixLength,
                prefix NSAPpreifx,
                qOSlgth QOSlength,
                qOSval QOSvalue }
QOSvalue ::= OCTET STRING(SIZE(1..255))
Rdi ::=OCTET STRING(SIZE(1..20))
  --assigned from the NSAP address space
Rdcgroup::=SEQUENCE{confed Rdcsetid, members SET OF Rdi}
Rdcsetid ::=INTEGER(1..255)
RDTransitDelay ::=INTEGER(0..65535)
Rdlre ::=INTEGER(0..4294967295)
RetransmissionTime ::= INTEGER(0..65535)
RemoteBIS-NET ::=NetworkEntityTitle
RemoteRDCConfig ::=Rdcgroup
RemoteBISNetActionReply ::=SEQUENCE{
   responseCode OBJECT IDENTIFIER,
   responseArgs SET OF Parameter OPTIONAL}
RibAttsSet ::= SEQUENCE  { confed Ribsetid,
   count Ribsetcount,
   attribs SET OF Ribattributes}
Ribsetid ::=INTEGER(1..255)
Ribsetcount ::=INTEGER(0..255)
Ribattributes ::= SEQUENCE {
             priority [0] EXPLICIT Priority OPTIONAL,
             security [1] EXPLICIT SEC OPTIONAL,
             qosmaint [2] EXPLICIT QOS OPTIONAL }
Ribattribute ::= ENUMERATED {
  tRANSITDELAY (9),
  rESIDUALERROR (10),
  eXPENSE (11),
  sourceSpecificQOS (12),
  destinationSpecificQOS (13),
```

```
  sourceSpecificSecurity (17),
  destinationSpecificSecurity(18),
  capacity (19),
  priority (20)}

Ribvalue ::= SEQUENCE {length Ribattlength,
   attr Ribattributes}
Ribattlength ::= INTEGER
Ribattvalue ::= CHOICE {
   transitdelayvalue [0] IMPLICIT INTEGER,
   residualerrorvalue [1] IMPLICIT INTEGER,
   expensevalue [2] IMPLICIT INTEGER,
   sourcespecificqos [3]IMPLICIT INTEGER,
   destinationspecificqos [4] IMPLICIT INTEGER,
   sourcespecificsecurity [5] IMPLICIT INTEGER,
   destinationspecificsecurity [6] IMPLICIT INTEGER,
   capacityvalue [7] IMPLICIT INTEGER,
   priorityvalue [8] IMPLICIT INTEGER}
Ribattqos ::= SEQUENCE {
   preflgth NSAPprefixLength,
   prefix NSAPprefix,
   qOSlgth QOSlength,
   qOSval QOSvalue,
   metriclgth Metriclength,
   metricval Metricvalue}
Ribattsec ::= SEQUENCE {
   preflgth NSAPprefixLength,
   prefix NSAPprefix,
   seclgth Securitylength,
   secval Securitylevel}
RouteAdvertisementInterval ::=INTEGER(30..900)
 --IS 10589 imposes minimum value of 30 seconds
 --and maximum value of 900 seconds in clause
 --12.2.3.4, part c)
SEC ::= CHOICE { ssSEC[0] EXPLICIT Ribattsec,
            dsSEC[1] EXPLICIT Ribattsec }
Securitylength ::= INTEGER(0..255)
Securitylevel ::= OCTET STRING(SIZE(1..255))

SNPAaddress ::= OCTET STRING
  (FROM ('1'H|'2'H|'3'H|'4'H|'5'H|'6'H|'7'H|'8'H|'9'H|
       'A'H|'B'H|'C'H|'D'H|'E'H|'F'H))
  --integral number of hexadecimal digits
SNPAaddresses ::= SET OF SNPAaddress
State ::= ENUMERATED {
  closed (0),
  open-recv(1),
  established(2),
  open-sent(3),
  close-wait(4)}
StopEventreply ::= Parameter
StartEventreply::=Parameter
SystemIdGroup ::= SEQUENCE  {
  nETS SET OF NETprefix,
  nSAPs SET OF ESprefix }
Updateerrorsubcode ::=ENUMERATED {
  malformedAttributelist (1),
  unrecognizedWell-knownAttribute (2),
  missingWell-knownAttribute (3),
  attributeFlagsError (4),
```

```
    attributeLengthError (5),
    rDRouteingLoop (6),
    invalidNEXTHOPAttribute (7),
    optionalAttributeerror (8),
    invalidReachabilityInformation (9),
    misconfiguredRDCs (10)}
  Version ::=INTEGER (1..255)
  Waitdelattimer ::= Timer
  END
```

12. Conformance

A Protocol Implementation Conformance Statement (PICS) shall be completed with respect to any claim for conformance of an implementation to this International Standard. The PICS shall be produced in accordance with the relevant PICS-proforma in Annex A.

NOTE 36: Since it is only Boundary ISs that implement the elements of procedure of this international standard, this clause does not address deployment guidelines or addressing guidelines for systems in general. Since distribution of policies is outside the scope of this standard, this topic also is not addressed in the following conformance clauses.

12.1 Static Conformance for All BISs

Each IS claiming conformance to this international standard shall be capable of each of the following:

a) generating and parsing BISPDUs with the following structure and formats described in: 6.1, 6.2, 6.3, 6.4, 6.5, 6.6, and 6.7

b) transmitting and receiving NSAP prefixes that have been encoded as single values according to 7.1.2.1

c) generating, recognizing upon receipt, and updating each of the following path attributes as described in the indicated clauses:

- ROUTE_SEPARATOR in 7.12.1
- RD_PATH in 7.12.3
- RD_HOP_COUNT in 7.12.4
- CAPACITY in 7.12.15

d) recognizing upon receipt each of the following well-known discretionary path attributes that are contained in any incoming UPDATE PDU, as described in the indicated clauses:

- EXT_INFO in 7.12.2
- NEXT_HOP in 7.12.4
- DIST_LIST_INCL in 7.12.5
- DIST_LIST_EXCL in 7.12.6
- TRANSIT DELAY in 7.12.8
- RESIDUAL ERROR in 7.12.9
- EXPENSE in 7.12.10
- LOCALLY DEFINED QOS in 7.12.11
- HIERARCHICAL RECORDING in 7.12.12
- SECURITY in 7.12.14
- CAPACITY in 7.12.15
- PRIORITY in 7.12.16

e) utilizing domain configuration information in the format described in 7.3

f) providing reliable delivery of BISPDUs using sequence numbering and flow control as described in 7.7.4 and 7.7.5.

g) establishing, closing, and maintaining BIS-BIS connections in accordance with the procedures of 7.6

h) responding to error conditions in BISPDUs according to 7.20

i) negotiating the protocol version number according to 7.8

j) operating in a ″fail-stop″ manner in regard to corrupted routeing information, according to 7.10.2

k) maintaining RIBs as described in 7.10.

l) handling incoming BISPDUs as described in 7.14.

m) detecting inconsistent routeing information in accordance with 7.15.1.

n) receiving and recognizing information which has been reduced in size according to the methods of 7.18.1.

o) distributing network reachability information within a routeing domain according to 7.17.1.

p) distributing network reachability information outside a routeing domain according to 7.17.2.

q) selecting routes according to 7.16

r) forwarding ISO 8473 NPDUs according to 8.

s) supporting the interface to ISO 8473 using the service primitives according to 9.

t) providing the managed objects described in 11.

12.2 Conformance to Optional Functions

12.2.1 Generation of Information in Reduced Form

A BIS that claims to support generation of information in a reduced form shall be capable of producing information in accordance with the reduction techniques of clauses 7.18.1 and 7.18.2.

12.2.2 Generation of Well-known Discretionary Attributes

A BIS that claims to support generation of any of the following well-known discretionary attributes shall do so in accordance with the indicated clauses:

ROUTE_SEPARATOR 7.12.1

EXT_INFO: 7.12.2

NEXT HOP: 7.12.4

DIST LIST INCL: 7.12.5

DIST LIST EXCL: 7.12.6

TRANSIT DELAY: 7.12.8

RESIDUAL ERROR: 7.12.9.

EXPENSE: 7.12.10.

LOCALLY DEFINED QOS: 7.12.11.

HIERARCHICAL RECORDING: 7.12.12

SECURITY: 7.12.14

CAPACITY: 7.12.15

PRIORITY: 7.12.16

12.2.3 Propagation of Well-known Discretionary Attributes

A BIS that claims to support updating and propagation of any of the following well-known discretionary attributes after having received them in an UPDATE PDU shall do so in accordance with the indicated clauses:

ROUTE_SEPARATOR 7.12.1

EXT_INFO: 7.12.2

NEXT HOP: 7.12.4

DIST LIST INCL: 7.12.5

DIST LIST EXCL: 7.12.6

TRANSIT DELAY: 7.12.8

RESIDUAL ERROR: 7.12.9.

EXPENSE: 7.12.10.

LOCALLY DEFINED QOS: 7.12.11.

HIERARCHICAL RECORDING: 7.12.12

SECURITY: 7.12.14

CAPACITY: 7.12.15

PRIORITY: 7.12.16

12.2.4 Peer Entity Authentication

A BIS that claims to support peer entity authentication shall do so in accordance with 7.7.2.

Annex A. PICS Proforma

(Normative)

A.1 Introduction

The supplier of a protocol implementation which is claimed to conform to International Standard XXX shall complete the applicable Protocol Implementation Conformance Statement (PICS) proforma.

A completed PICS proforma is the PICS for the implementation in question. The PICS is a statement of which capabilities and options of the protocol have been implemented. The PICS can have a number of uses, including use:

— by the protocol implementer, as a check list to reduce the risk of failure to conform to the standard through oversight;

— by the supplier and acquirer—or potential acquirer— of the implementation, as a detailed indication of the capabilities of the implementation, stated relative to the common basis of understanding provided by the standard PICS proforma;

— by the user—or potential user— the implementation, as a basis for initially checking the possibility of interworking unit another implementation (note that, while interworking can never be guaranteed, failure to interwork can often be predicted from incompatible PICSs);

— by a protocol tester, as the basis for selecting appropriate tests against which to assess the claim for conformance of the implementation.

A.2 Abbreviations and Special Symbols

A.2.1 Status Symbols

The following status symbols are used in the PICS:

Symbol	Meaning
M	mandatory
O	optional
O.<n>	optional, but support of at least one of the group of options labelled by the same numeral <n> is required
X	prohibited
c.<cid>	conditional requirement, according to the condition identified by <cid>
<item>	simple-predicate condition, dependent on the support marked for <item>
—	not applicable (N/A)

A.3 Instructions for Completing the PICS Proforma

A.3.1 General Structure of the PICS Proforma

The first part of the PICS proforma—Implementation Identification and Protocol Summary—is to be completed as indicated with the information necessary to identify fully both the supplier and the implementation.

The main part of the PICS proforma is a fixed-format questionnaire divided into subclauses, each containing a group of individual items. Answers to the questionnaire items are to be provided in the rightmost column, either by simply marking an answer to indicate a restricted choice (usually *Yes* or *No*), or by entering a value or a set or range of values. (Note that there are some items where two or more choices from a set of possible answers can apply: all relevant choices are to be marked.)

Each item is identified by an item reference in the first column; the second column contains the question to be answered; the third column contains the reference or references to the material that specifies the item in the main body of the standard; the remaining columns record the status of the item—whether support is mandatory, optional, or conditional—and provide space for the answers: see also A.3.4 below.

A supplier may also provide—or be required to provide—further information, categorized as either Additional Information or Exception Information. When present, each kind of further information is to be provided in a further subclause of items labelled A<i> or X<i>, respectively, for cross-referencing purposes, where <i> is any unambiguous identification for the item (e.g., simply a numeral): there are no other restrictions on its format and presentation.

A completed PICS proforma, including any Additional Information and Exception Information, is the Protocol Implementation Conformance Statement for the implementation in question.

NOTE 37: Where an implementation is capable of being configured in more than one way, a single PICS may be able to describe all such configurations. However, the supplier has the choice of providing more than one PICS, each covering some subset of the implementation's configuration capabilities, in case that makes for easier and clearer presentation of the information.

A.3.2 Additional Information

Items of Additional Information allow a supplier to provide further information intended to assist the interpretation of the PICS. It is not intended or expected that a large quantity will be supplied, and a PICS can be considered complete without any such information. Examples might be an outline of the ways in which a (single) implementation can be set up to operate in a variety of environments and configurations, or a brief rationale—based perhaps upon specific application needs—for the exclusion of features which, although optional, are nonetheless commonly present in implementations of the protocol.

References to items of Additional Information may be entered next to any answer in the questionnaire, and may be included in items of Exception Information.

A.3.3 Exception Information

It may occasionally happen that a supplier will wish to answer an item with mandatory or prohibited status (after any conditions have been applied) in a way that conflicts with the indicated requirement. No preprinted answer will be found in the Support column for this: instead, the supplier is required to write into the Support column an X<i> reference to an item of Exception Information, and to provide the appropriate rationale in the Exception item itself.

An implementation for which an Exception item is required in this way does not conform to ISO/IEC XXX.

NOTE 38: A possible reason for the situation described above is that a defect in the standard has been reported, a correction for which is expected to change the requirement not met by the implementation.

A.3.4 Conditional Status

A.3.4.1 Conditional Items

The PICS proforma contains a number of conditional items. These are items for which the status that applies—mandatory, optional, or prohibited—is dependent upon whether or not certain other items are supported, or upon the values supported for other items.

In many cases, whether or not the item applies at all is conditional in this way, as well as the status when the item does apply.

Individual conditional items are indicated by a conditional symbol in the Status column as described in A.3.4.2 and A.3.4.3 below. Where a group of items is subject to the same condition for applicability, a separate preliminary question about the condition appears at the head of the group, with an instruction to skip to a later point in the questionnaire if the "Not Applicable" answer is selected.

A.3.4.2 Conditional Symbols and Conditions

A conditional symbol is either of the form C.<n> or C.G<n>, where <n> is a numeral or is an abbreviated condition of the form described in A.3.4.3 below. For the C.<n> form, the numeral identifies a condition appearing in a list as the end of the subclause containing the item. For the C.G<n> form, the numeral identifies a condition appearing in a list of global conditions.

A simple condition is of the form

```
if <p1> then <s1> else <s2>
```

where <p1> is a predicate (see A.3.4.4 below) and <s1> and <s2> are either basic status symbols (M, O, O.<n>, or X) or the symbol "—".

An extended condition is of the form

```
if <p1> then <s1>
else if <p2> then <s2>
[else if ...]
else <sn>
```

where the quantities <px> are predicates, and the quantities <sx> are either basic status symbols or "—".

The symbol (either a basic status symbol or "—") applicable to an item governed by a simple condition is <s1> if the predicate of the condition is true, and is <s2> otherwise. The symbol applicable to an item governed by an extended condition is <si> where <pi> is the first true predicate, if any, in the sequence <p1>, <p2>,...; and it is <sn> if no predicate is true.

A.3.4.3 Abbreviated Conditions

The abbreviated condition <item>:<s> in the status column is equivalent to a conditional symbol with corresponding condition *if <item> then <s> else "—"*.

A.3.4.4 Predicates

A simple predicate in a condition is either:

a) a single item reference; or

b) a relation containing a comparison operator (=,<,etc.) with one (or both) of its operands being an item reference for an item taking numerical values as its answer. In case (a), the predicate is true if the item referred to is marked as supported, and false otherwise. In case (b), the

Item	Questions/Features	References	Status	N/A	Support
H3	Is ... supported?	42.3(d)	C.2	__	M: Yes __ O: Yes__ No__

C.2: If A4 then M else if D1 or (B52>2) then O else N/A

Figure 9. Illustrative PICS for Conditional Items

predicate is true if the relation holds when each item reference is replaced by the value entered in the Support column as the answer to the item referred to.

Compound predicates are boolean expressions constructed by combining simple predicates using the boolean operators AND, OR, and NOT, and parentheses, in the usual way. A compound predicate is true if and only if the boolean expression evaluates to ″true″ when the simple predicates are interpreted as described above.

Items whose references are used in predicates are indicated by an asterisk in the Item column.

A.3.4.5 Answering Conditional Items

To answer a conditional item, the predicate(s) of the condition is (are) evaluated as described in A.3.4.4 above, and the applicable symbol is determined as described in A.3.4.2. If the result is ″N/A″, the Not Applicable answer column is to be marked; otherwise, the Support column is to be completed in the usual way.

When two or more status symbols appear in the condition for an item, the Support column for the item contains one line for each such symbol, labelled by the relevant method. The answer for the item is to be marked in the line labelled by the symbol selected according to the condition (unselected lines may be crossed out for added clarity).

For example, in Figure 9, the N/A column would be marked if neither predicate of C.2 was true; the answer line labelled ″M:″ would be marked if item A4 was marked as supported; and the answer line labelled ″O:″ would be marked it item A4 was not marked as supported but item D1 was supported.

A.4 Identification

A.4.1 Implementation Identification: IDRP

Table 7. PICS Proforma for IDRP: Implementation Identification	
Supplier	
Contact point for queries about this PICS	
Implementation Name(s) and Version(s)	
Other information necessary for full identification (e.g., Name′s and Version(s) for machines and operating systems, System Name(s))	

NOTE 39: Only the first three items are required for all implementations; other information may be completed as appropriate in meeting the requirement for full identification. The terms *Name* and *Version* should be interpreted appropriately to correspond with a supplier′s terminology (using, e.g., Type,Series, MODEL).

A.4.2 Protocol Summary: IDRP

Table 8. PICS Proforma for IDRP: Protocol Summary	
Protocol Version	
Addenda Implemented (if applicable)	
Amendments Implemented	
Have any Exception items been required? (See A.3.3.)	Yes__ No__ **NOTE:** The answer *Yes* means that the implementation does not conform to this international standard.)

A.4.3 PICS Proforma: IDRP General

Table 9. PICS Proforma for IDRP: General				
Item	**Questions/Features**	**References**	**Status**	**Support**
BASIC	Are all basic BIS functions implemented?	12.1	M	Yes __
MGT	Is this system capable of being managed by the specified management information?	11 .	M	Yes __
VER	Does this BIS support version negotiation?	7.8	M	Yes__
RTSEP	Does this BIS support the ROUTE_SEPARATOR attribute?	7.12.1	M	Yes__
HOPS	Does this BIS support the RD_HOP_COUNT attribute?	7.12.13	M	Yes__
PATH	Does this BIS support the RD_PATH attribute?	7.12.3.	M	Yes__
CAPY	Does this BIS support the CAPACITY attribute?	7.12.15	M	Yes__
FSM	Does this BIS manage BIS-BIS connections according to the BIS FSM description?	7.6.1	M	Yes__
FCTL	Does this BIS provide flow control?	7.7.5	M	Yes__
SEQNO	Does this BIS provide sequence number suport?	7.7.4	M	Yes__
INTG1	Does this BIS provide data integrity using authentication type 1?	7.7.1	O.1	Yes__ No__
INTG2	Does this BIS provide data integrity using authenticationtype 2?	7.7.2	O.1	Yes__ No__
INTG3	Does this BIS provide data integrity using authentication type 3?	7.7.3	O.1	Yes__ No__
ERROR	Does this BIS handle error handling for IDRP?	7.20	M	Yes__
RIBCHK	Does this BIS operate in a ″fail-stop″ manner with respect to corrupted routeing information?	7.10.2	M	Yes__

A.4.4 PICS Proforma: IDRP Update Send Process

Table 10. PICS Proforma for IDRP: Update-Send				
Item	**Questions/Features**	**References**	**Status**	**Support**
INT	Does this BIS provide the internal update procedures?	7.17.1	M	Yes __
RTSEL	Does this BIS support the **MinRouteSelectionInterval** timer?	7.17.3.1	M	Yes __
RTORG	Does this BIS support the **MinRDOriginationInterval** timer?	7.17.3.2	M	Yes __
JITTER	Does this BIS provide jitter on its timers?	7.17.3.3	M	Yes __

A.4.5 PICS Proforma: IDRP Update Receive Process

Table 11. PICS Proforma for IDRP: Update-Receive				
Item	**Questions/Features**	**References**	**Status**	**Support**
INPDU	Does the BIS handle inbound BISPDUs correctly?	7.14	M	Yes __
INCONS	Does this BIS detect inconsistent routeing information?	7.15.1	M	Yes__

A.4.6 PICS Proforma: IDRP Decision Process

Table 12. PICS Proforma for IDRP: Decision				
Item	**Questions/Features**	**References**	**Status**	**Support**
TIES	Does the BIS break ties between candidate routes correctly?	7.16.2.1	M	Yes __
RIBUPD	Does this BIS update the correct Loc-RIBs?	7.16.2	M	Yes__
AGGRT	Does this BIS support route aggregation?	7.18.2.1, 7.18.2.2, 7.18.2.3	O	Yes__ No __
LOCK	Does this BIS provide interlocks between its decision process and the updating of the information in its Adj-RIBs-In?	7.16.4	M	Yes__

A.4.7 PICS Proforma: IDRP Receive Process

Table 13. PICS Proforma for IDRP: Receive				
Item	**Questions/Features**	**References**	**Status**	**Support**
RCV	Does the BIS process incoming BISPDUs and respond correctly to error conditions?	7.14, 7.20	M	Yes __
OSIZE	Does the BIS accept incoming OPEN PDUs whose size in octets is between **minBISPDULength** and 3000?	6.2, 7.20	M	Yes __
MXPDU	Does the BIS accept incoming UPDATE, IDRP ERROR and RIB REFRESH PDUs whose size in octets is between **minBISPDULength** and **maxBISPDULength**?	6.2, 7.20	M	Yes __

A.4.8 PICS Proforma: IDRP CLNS Forwarding

Table 14. PICS Proforma for IDRP: CLNS Forwarding				
Item	**Questions/Features**	**References**	**Status**	**Support**
PSRCRT	Does the BIS correctly handle 8473 NPDUs that contain a partial source route?	8	M	Yes __
DATTS	Does the BIS correctly extract the NPDU-derived Distinguishing Attributes from an 8473 NPDU?	8.2	M	Yes __
MATCH	Does the BIS correctly match the NPDU-derived Distinguishing Attributes with the corresponding FIB-Atts?	8.3	M	Yes __
EXTF	Does the BIS correctly forward NPDUs with destinations outside its own routeing domain?	8.4	M	Yes __
INTF	Does the BIS correctly forward NPDUs with destinations inside its own routeing domain?	8.1	M	Yes __

A.4.9 PICS Proforma: IDRP Authentication

Table 15. PICS Proforma for IDRP: Peer Entity AUthentication				
Item	**Questions/Features**	**References**	**Status**	**Support**
AUTH	Does the BIS correctly authenticate the source of a BISPDU?	7.7.2	O	Yes__ No __

A.4.10 PICS Proforma: IDRP Optional Transitive Attributes

Table 16. PICS Proforma for IDRP: Optional Transitive Attributes				
Item	**Questions/Features**	**References**	**Status**	**Support**
MEXIT	Does the BIS support use of the MULTI-EXIT DISC attribute?	7.12.7	O	Yes __ No __

A.4.11 PICS Proforma: IDRP Well Known Discretionary Attributes

Table 17. PICS Proforma for IDRP: Generating Well Known Discretionary Attributes				
Item	**Questions/Features**	**References**	**Status**	**Support**
EXTG	Does the BIS support generation of the EXT_INFO attribute?	7.12.2	O	Yes __ No __
NHRS	Does the BIS support generation of the NEXT_HOP attribute in support of route servers?	7.12.4	O	Yes __ No __
NHSN	Does the BIS support generation of the NEXT_HOP attribute to advertise SNPAs?	7.12.4	O	Yes __ No __
DLI	Does the BIS support generation of the DIST_LIST_INCL attribute?	7.12.5	O	Yes __ No __
DLE	Does the BIS support generation of the DIST_LIST_EXCL attribute?	7.12.6	O	Yes __ No __
TDLY	Does the BIS support generation of the TRANSIT DELAY attribute?	7.12.8	O	Yes __ No __
RERR	Does the BIS support generation of the RESIDUAL ERROR attribute?	7.12.9	O	Yes __ No __
EXP	Does the BIS support generation of the EXPENSE attribute?	7.12.10	O	Yes __ No __
LQOSG	Does the BIS support generation of the LOCALLY DEFINED QOS attribute?	7.12.11	O	Yes __ No __
HREC	Does the BIS support generation of the HIERARCHICAL RECORDING attribute?	7.12.12	O	Yes __ No __
SECG	Does the BIS support generation of the SECURITY attribute?	7.12.14	O	Yes __ No __
PRTY	Does the BIS support generation of the PRIORITY attribute?	7.12.16	O	Yes __ No __

Table 18. PICS Proforma for IDRP: Propagating Well Known Discretionary Attributes				
Item	**Questions/Features**	**References**	**Status**	**Support**
EXTGP	Does the BIS support propagation of the EXT_INFO attribute?	7.12.2	M	Yes __ No __
NHRSP	Does the BIS support propagation of the NEXT_HOP attribute in support of route servers?	7.12.4	O	Yes __ No __
NHSNP	Does the BIS support propagation of the NEXT_HOP attribute to advertise SNPAs?	7.12.4	O	Yes __ No __
DLIP	Does the BIS support propagation of the DIST_LIST_INCL attribute?	7.12.5	O	Yes __ No __
DLEP	Does the BIS support propagation of the DIST_LIST_EXCL attribute?	7.12.6	O	Yes __ No __
TDLYP	Does the BIS support propagation of the TRANSIT DELAY attribute?	7.12.8	O	Yes __ No __
RERRP	Does the BIS support propagation of the RESIDUAL ERROR attribute?	7.12.9	O	Yes __ No __
EXPP	Does the BIS support propagation of the EXPENSE attribute?	7.12.10	O	Yes __ No __
LQOSP	Does the BIS support propagation of the LOCALLY DEFINED QOS attribute?	7.12.11	O	Yes __ No __
HRECP	Does the BIS support propagation of the HIERARCHICAL RECORDING attribute?	7.12.12	O	Yes __ No __
SECP	Does the BIS support propagation of the SECURITY attribute?	7.12.14	O	Yes __ No __
PRTYP	Does the BIS support propagation of the PRIORITY attribute?	7.12.16	O	Yes __ No __

Table 19. PICS Proforma for IDRP: Receiving Well Known Discretionary Attributes				
Item	**Questions/Features**	**References**	**Status**	**Support**
EXTR	Does the BIS recognize upon receipt the EXT_INFO attribute?	7.12.2	M	Yes __ No __ X:
NHRSR	Does the BIS recognize upon receipt the NEXT_HOP attribute?	7.12.4	M	Yes __ No __ X:
DLIR	Does the BIS recognize upon receipt the DIST_LIST_INCL attribute?	7.12.5	M	Yes __ No __ X:
DLER	Does the BIS recognize upon receipt the DIST_LIST_EXCL attribute?	7.12.6	M	Yes __ No __ X:
TDLYR	Does the BIS recognize upon receipt the TRANSIT DELAY attribute?	7.12.8	M	Yes __ No __ X:
RERRR	Does the BIS recognize upon receipt the RESIDUAL ERROR attribute?	7.12.9	M	Yes __ No __ X:
EXPR	Does the BIS recognize upon receipt the EXPENSE attribute?	7.12.10	M	Yes __ No __ X:
LQOSR	Does the BIS recognize upon receipt the LOCALLY DEFINED QOS attribute?	7.12.11	M	Yes __ No __ X:
HRECR	Does the BIS recognize upon receipt the HIERARCHICAL RECORDING attribute?	7.12.12	M	Yes __ No __ X:
SECR	Does the BIS recognize upon receipt the SECURITY attribute?	7.12.14	M	Yes __ No __ X:
PRTYR	Does the BIS recognize upon receipt the PRIORITY attribute?	7.12.16	M	Yes __ No __ X:

Annex B. Bibliography

(Normative)

The following references contain information which is helpful in understanding the protocol described in this international standard, and for setting the context in which it might be deployed.

ISO 9542:1988, *Information Processing Systems - Telecommunications and Information Exchange between Systems - End system to Intermediate system routeing exchange protocol for use in conjunction with the Protocol for providing the connectionless-mode network service (ISO 8473)*

ISO TR 9575: 1989, *Information Processing Systems - Telecommunications and Information Exchange between Systems - OSI Routeing Framework*

ISO/IEC 10589, *Information Processing Systems - Telecommunications and Information Exchange between Systems - Intermediate System to Intermediate System Intra-Domain Routing Exchange Protocol for use in Conjunction with the protocol for providing the Connectionless-mode Network Service (ISO 8473)*

ISO/IEC CD 11577, *Information Technology - Telecommunications and Information Exchange between Systems - Network Layer Security Protocol*

RFC 1186, *The MD4 Message Digest Algorithm*, R. Rivest, October 1990

Annex C. IDRP Checksum Generation Algorithm

(Normative)

This annex describes the IDRP checksum algorithm, which accepts an a message of arbitrary length as its input and produces a 128-bit digital signature as its output. It is based upon the message digest algorithm described in RFC 1186.

C.1 Mathematical Notation

In this annex, the following notation is used:

Symbol	Meaning
X+**Y**	Addition of two quantities, modulo 2^{32}

X«s	Left rotation (circular shifting) of the binary pattern X by "s" bit positions.
¬ X	Bitwise complement of the binary pattern X
X ⊕ Y	Bitwise EXCLUSIVE-OR function of X and Y
XY	Bitwise AND-function of X and Y
X ∨ Y	Bitwise OR-function of X and Y

C.2 Algorithm Description

The input data stream, *M*, operated upon by this algorithm is assumed to be *b* binary digits in length. The first (leftmost) bit of *M* is labelled m_1, the second is labelled m_2, ..., and the last (rightmost) bit is labelled m_b.

The following steps shall be performed to compute the message digest of the message:

a) *Append padding bits*

From 1 to 512 padding bits shall be appended to the back of the original message *M* so that its length in bits is congruent to 448, modulo 512. If the original message length, *b* is already congruent to 448 modulo 512, then 512 bits of padding shall be added. The first padding bit shall be 1, and all others shall be 0.

b) *Append the length field*

When the value of *b* is less than or equal to 2^{64}, it shall be expressed as a 64-bit binary integer. If the quantity *b* is greater than 2^{64}, then only the low-order 64 bits of its binary representation shall be used. The 64-bit long binary encoded quantity shall then be appended to the back of the result obtained in the first step. Call this quantity *Q*.

After completing these two steps, the quantity *Q* will have a length which is an exact multiple of 512 bits. That is, *Q* consists of *N* 32-bit words, where *N* is a multiple of 16. Let *Q*[1] represent the first (leftmost) 32-bit word of *Q*,..., and *Q*[*N*] represent the last (rightmost) 32-bit word of *Q*.

c) *Initialize the Checksum Buffer*

The checksum is accumulated in 4 32-bit buffers (A, B, C, and D). Each shall be initialized to the following values, expressed in hexadecimal notation:

Word A initial value: 01 23 45 67

Word B initial value: 89 AB CD EF

Word C initial value: FE DC BA 98

Word D initial value: 76 54 32 10

d) *Process Q in Blocks of 16 32-bit words*

Three auxiliary functions are defined that each take three 32-bit words as input and produce one 32-bit word as output;

$f(X,Y,Z) = XY \vee (\neg X)Z$

$g(X,Y,Z) = XY \vee XZ \vee YZ$

$h(X,Y,Z) = X \oplus Y \oplus Z$

Do the following:

```
For i = 0 to N/16 do  /* process each 16-word block */
  For j = 1 to 16 do: /* copy block i into X */
     set X[j] to M[i*16+j].
  end /* of loop on j */
  Save A as AA, B as BB, C as CC, and D as DD.
```

[Round 1]:

Let [K L M P t s] denote the operation
K = (K+ f(L,M,P) + X[t]) « s .

Do the following 16 operations in the order indicated:

```
[A B C D 0 3]
[D A B C 1 7]
[C D A B 2 11]
[B C D A 3 19]
[A B C D 4 3]
[D A B C 5 7]
[C D A B 6 11]
[B C D A 7 19]
[A B C D 8 3]
[D A B C 9 7]
[C D A B 10 11]
[B C D A 11 19]
[A B C D 12 3]
[D A B C 13 7]
[C D A B 14 11]
[B C D A 15 19]
```

[Round 2]:

Now let [K L M P t s] denote the operation
K = (K + g(L,M,P) + X[t] + 5A827999) «s .

(The value 5A827999 is a hexadecimal 32-bit constant.)

Do the following 16 operations in the order indicated:

[A B C D 0 3]
[D A B C 4 5]
[C D A B 8 9]
[B C D A 12 13]
[A B C D 1 3]
[D A B C 5 5]
[C D A B 9 9]
[B C D A 13 13]
[A B C D 2 3]
[D A B C 6 5]
[C D A B 10 9]
[B C D A 14 13]
[A B C D 3 3]
[D A B C 7 5]
[C D A B 11 9]
[B C D A 15 13]

[Round 3]:

Now let [K L M P t s] denote the operation
K = (K + h(L,M,P) + X[t] + 6ED9EBA1)«s.

(The value 6ED9EBA1 is a hexadecimal 32-bit constant.)

Do the following 16 operations in the order indicated:

[A B C D 0 3]
[D A B C 8 9]
[C D A B 4 11]
[B C D A 12 15]
[A B C D 2 3]
[D A B C 10 9]
[C D A B 6 11]
[B C D A 14 15]
[A B C D 1 3]
[D A B C 9 9]
[C D A B 5 11]
[B C D A 13 15]
[A B C D 3 3]
[D A B C 11 9]
[C D A B 7 11]
[B C D A 15 15]

Then perform the following additions:

A = A + AA
B = B + BB
C = C + CC
D = D + DD

(That is, each register is incremented by the value it had when processing on this block was started.)

end /* of loop on i */

e) *Output*

After completing the last loop on *i*, the checksum is the concatenation of the final values of A, B, C, and D.

Annex D. Example of Authentication Type 2

(Informative)

D.1 Authentication Mechanism

The procedure outlined below provides data integrity and peer BIS authentication in a way that satisfies the requirements of authentication type 2. This is an illustrative example only. Any other method that is consistent with type 2 authentication could also be used.

For an OPEN PDU with an authentication code field of 2, and for all BISPDUs that flow on a BIS-BIS connection established by this OPEN PDU, the validation field will contain a 16-octet encrypted checksum:

a) Generating a Validation Pattern:

The contents of the Validation Pattern field that is included in an outbound BISPDU can be generated by the following two step process, which is illustrated in Figure 10:

1) An unencrypted checksum that covers the contents of the BISPDU can be generated by applying the procedures of Annex C to the input data stream that consists of the contents of the entire BISPDU with all bits of the Validation Pattern field initially set to 0. The output of this step is an unencrypted 16-octet long checksum, which is called *chksum*.

2) The 16-octet quantity *chksum* is then encrypted, and the encrypted pattern is placed in the Validation Pattern field of the BISPDU.

NOTE 40: The following observations can be made:

1) The encryption algorithm must be agreed upon in the cryptographic association set up by the two BISs involved in the authentication process. This international standard does not mandate use of a specific encryption algorithm. Explicit indication of the specific algorithm to be used is outside the scope of IDRP. However, the "Authentication Data" field of IDRP's OPEN PDU can be used to specify an algorithm indirectly in accordance with the local agreements of the two communicating BISs.

2) There is no requirement that a given BIS must use the same encryption algorithm on every BIS-BIS connection which it has

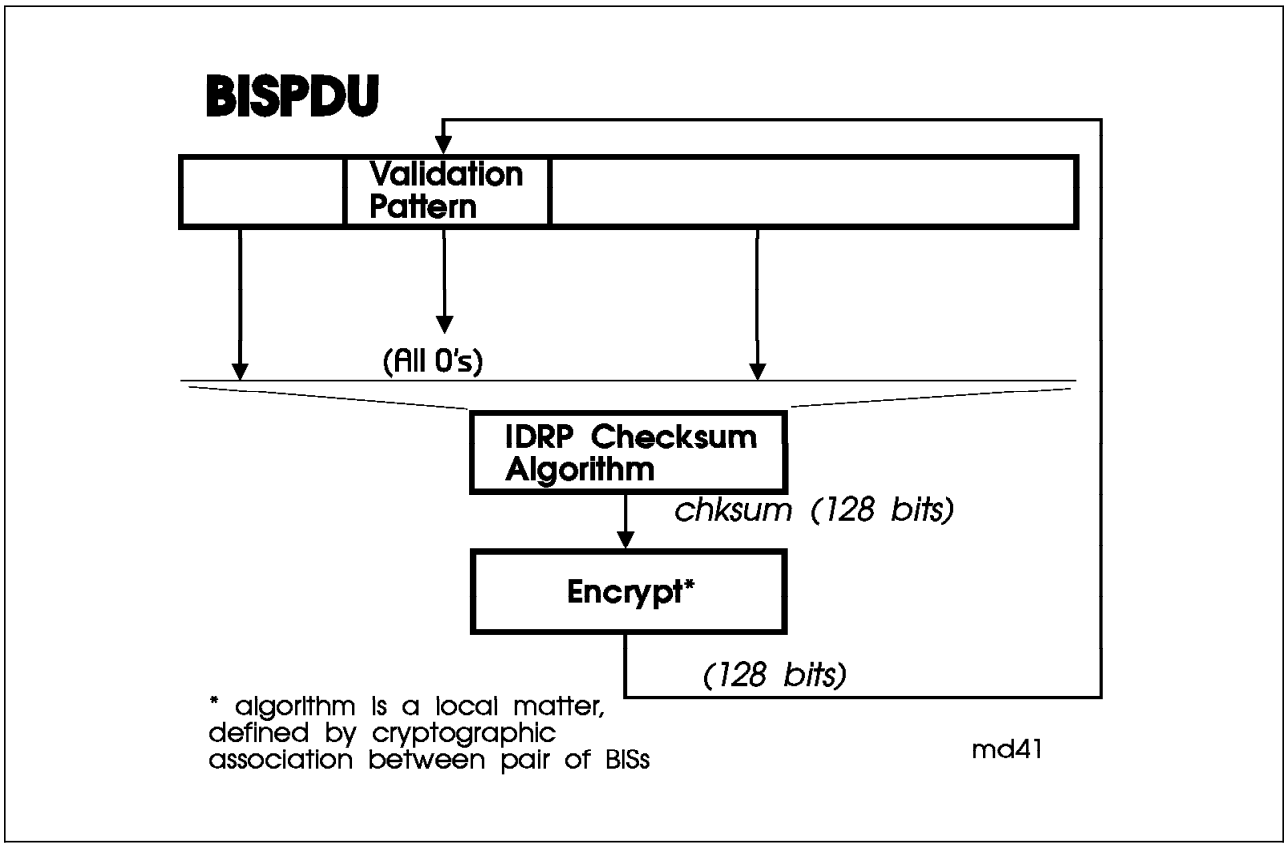

Figure 10. An Example of the Authentication Type 2

established. The IDRP authentication code carried in the OPEN PDU applies only to a particular BIS-BIS connection; Thus, different BIS-BIS connections may choose to use different encryption algorithms.

3) The presence or absence of the authentication function is specified on a "per BIS-BIS connection" basis. Thus, a given BIS may support some BIS-BIS connections that use authentication, and others that do not.

b) Checking the Validation Pattern of an Inbound BISPDU:

The contents of the Validation Pattern field of an inbound BISPDU will be checked by the following procedures:

1) Apply the IDRP checksum algorithm to the data stream that consists of the contents of the inbound BISPDU with its Validation Pattern set to all zeros. Call this quantity the "reference pattern".

2) Decrypt the Validation Pattern field of the inbound BISPDU, calling the result the "received pattern".

If the "reference pattern" and the "received pattern" are identical, then the peer BIS has been authenticated, and the inbound BISPDU will be accepted. If the "reference pattern" and the "received pattern" are not identical, the receiving BIS will inform system management that an authentication failure has occurred. The incoming BISPDU will be ignored. The receiving BIS will not send an IDRP ERROR PDU to the peer-BIS because the identity of the peer has not been authenticated.

NOTE 41: If a BISPDU has a malformed header, it will be discarded. As a result, the Validation Pattern of such BISPDUs will not be checked.

Annex E. Jitter Algorithm

(Informative)

When BISPDUs are transmitted as a result of timer expiration, there is danger that the timers of individual systems could become synchronized. To minimize the likelihood of this occurring, all periodic timers whose expiration can cause the transmission of a BISPDU shall have jitter introduced. An example algorithm that satisfies the requirements of clause 7.17.3.3. is shown in Figure 11.

```
CONSTANT
  Jitter=0.25 (* defined by architectural constant Jitter *)
  Resolution=100

PROCEDURE Random (max: Integer): Integer
  (* This procedure delivers a uniformly distributed random integer R,
   such that 0 < R < max *)

PROCEDURE
  DefineJitteredTimer (baseTimeValueinSeconds: Integer: expirationAction: Procedure);

VAR
  baseTimeValue, maximumTimeModifier, waitTime: Integer;
  nextexpirtation: Time;

BEGIN
  baseTimeValue:=baseTimeValueInSeconds*1000/Resolution;
  maximumTimeModifier:=baseTimeValue*Jitter;
  WHILE running DO
    BEGIN
    (* First compute next expiration timer *)
    randomTimeModifier:=Random(maximumTimeModifier);
    waitTime:=baseTimeValue - randomTimeModifier;
    waitTime:=waitTime*Resolution/1000;
    nextexpiration:=CurrentTime + waitTime;
    (* Then perform expiration action *)
    expirationAction: WaitUntil(nextexpiration)
    END (* of loop *)
 END (* of DefinedJitterTimer *)
```

Figure 11. Jitter Algorithm for Timers

Annex F. Computing a Checksum for an Adj-RIB

(Informative)

To compute the checksum for a given Adj-RIB-Out or Adj-RIB-In, the following procedure can be used:

a) Unfeasible routes will not enter into the computation.

b) A sequence number will be associated with each feasible route in the information base. For an Adj-RIB-Out, it will be the locally generated sequence number of the UPDATE PDU that was used to advertise the route; for an Adj-RIB-In, it will be the sequence number of the neighbor BIS's UPDATE PDU that advertised the route.

c) The feasible routes within the information base will be sorted in a non-decreasing order of their sequence numbers.

d) Within each route, path attributes will be sorted in a non-decreasing order based on their type codes, followed by the Network Layer Reachability Information sorted in lexicographical order, based on the binary value of its NSAP address prefixes.

e) The checksum will be generated by applying the procedures of Annex C to the octet stream which is composed from the concatenation of the sorted feasible routes.

Annex G. RIB Overload

(Informative)

A BIS is said to experience a RIB-overload condition when it does not have enough memory available to store the routeing information needed for its Adj-RIBs-In, Loc-RIBs, and Adj-RIBs-Out. Since the routeing information that a BIS chooses to maintain is in fact a local policy, this international standard does not prescribe methods for handling overload conditions.

Methods for handling RIB overload can be considered as specific instances of local policies, and therefore

are not specified by this international standard. However, some examples of approaches that may be used to control RIB overload are suggested below.

Since the Loc-RIBs contain only a subset of the routeing information held in the Adj-RIBs-In, the size of a BIS's Adj-RIBs-In will be greater than or equal to the size of its Loc-RIBs. Therefore, the first step to alleviate the memory overload condition would be to reduce the amount of information that is stored in Adj-RIBs-In. To insure routeing consistency within a routeing domain, the following steps can be applied to an Adj-RIB-In that is associated with a peer BIS located in an adjacent routeing domain:

a) Remove routes that are not currently in any of the Loc-RIBs (that is, those routes that have not been selected by the Decision Process):

   — Any routes to destinations that are not contained in the routes stored in the Loc-RIB may be removed with no negative impact.

   — Any routes to destinations that are contained in the routes stored in the Loc-RIBs can also be removed. However, since they could later have been used as fallback routes (if the current route that is in the Loc-RIB becomes unfeasible), removing them may cause sub-optimal connectivity in the future.

b) If several Adj-RIBs-In (that have the same RIB attribute but are associated with different neighbor BISs) have routes to the same destination, then routes with higher degree of preference (as computed by the local BIS) should be retained, while routes with lower degree of preference may be deleted.

c) If a BIS unilaterally deletes a route, then any solicited RIB-refresh will reinstate the deleted route. Hence, if the condition persists, the memory-overloaded BIS should close the IDRP connection, and then take corrective action, such as re-opening it with an OPEN PDU that indicates support for a smaller **RIB-ATTsSet**, for example.

d) Terminate one or more of the IDRP sessions with other BISs. That would result in releasing the memory that was previously used to store the Adj-RIB-Ins and the Adj-RIBs-Out associated with that BIS. To ensure routeing consistency within an RD this measure may be applied only to the IDRP sessions with BISs in adjacent RDs.

e) If all else fails to alleviate the memory overload condition, the local BIS can terminate all of its IDRP sessions.

Annex H. Processor Overload

(Informative)

A BIS is said to be CPU overloaded when there is not enough processing power to process incoming BISPDUs received from other BISs. In this situation BIS must continue to update the Adj-RIBs-In with information contained in BISPDUs received from other BISs, but may not run the Decision Process using this information except for routes identified in the WITHDRAWN ROUTES field of an UPDATE PDU.

If a route received in the UPDATE_PDU has the WITHDRAW path attribute, the local BIS checks whether this route is currently installed in one of its Loc-RIBs; if so, it removes it from the appropriate Loc-RIB, updates the appropriate Adj-RIBs-Out and FIB, and generates (if necessary) an UPDATE-PDU to inform other BIS's of the change in its Loc-RIBs and its Adj-RIBs-Out. The Decision Process on the local BIS does not select another to replace the one that becomes unfeasible.

Since this procedure decreases the size of the Loc-RIB, a long-lasting CPU overload condition can eventually deplete the entire Loc-RIB, thus making the BIS unavailable as an intermediate system. If the CPU overload condition disappears, then the Decision Process and Update Process should be run over all the new routes that were installed into the Adj-RIBs but have not yet been processed by the Decision Process. If the CPU overload condition persists for more than the predefined architectural constant **MaxCPUOverloadPeriod**, the local BIS terminates its IDRP sessions.

The order of termination of the IDRP sessions is significant. First the BIS should terminate one or more of the IDRP sessions with BISs in adjacent RDs. If after terminating IDRP sessions with all of the BISs in adjacent RDs the CPU overload still persists, the BIS terminates the rest of its IDRP sessions (with all the BISs within its own RD).

Annex I. Formation of RDCs

(Informative)

Confederations exist in the knowledge configured into a given BIS. Since this knowledge must be added one BIS at a time, it is necessary to examine how a confederation can grow, and what happens in the interim when only some of the BISs are aware of the information regarding the confederation.

There are some potential problems that one should be aware of:

— Routes through a confederation might not work properly if BISs in the middle of the confederation do not know about the confederation.

— Routes may not work properly while a planned very large confederation with confederations nested inside is growing, but is not yet large enough to include all the confederations that eventually will be nested inside.

— Policies in distant BIS's must change when confederations are formed or dissolve.

If confederations are formed and dissolved carefully, then these problems can be avoided. The next sections describe the steps that should be taken for several common scenarios.

I.1 Forming a New Lower Level Confederation

Let's start with the simplest case—a newly formed confederation consisting of several RDs. The steps involved are:

a) First warn all managers of all BISs whose RDs are contained in the new RDC that a new confederation will be formed consisting of the RDs in a particular set. If the new confederation is to be nested within an existing confederation, the existence of the new confederation will not be noticeable to any BISs outside the nesting confederation. Thus the affected BISs are those within the lowest level confederation in which this new confederation will be nested. If there are multiple overlapping confederations in which the confederation will be nested, with no smaller confederation nested within one of the overlapping confederations and in which the new confederation will be nested, BISs in all those confederations will be affected.

b) A manager of a BIS that has policies regarding any of the RDs to be included in the new confederation must modify those policies since the RDs in the new RDC can no longer be differentiated. For example, if the previous policy was that some of those were all right to route through and others not, a new single policy for the new confederation would need to be formulated.

c) The policy regarding the new confederation must be added to the existing set of policies (the confederation will not appear immediately).

d) When ample time has elapsed so that managers will can modify the policies at their BISs, the managers of the BISs in the new confederation can start informing them about the new confederation.

e) One by one, each BIS is informed that it is in the confederation. The order in which the BISs are modified is critical. At all times the set of BISs that have been informed about the confederation must be a connected set. Thus the confederation must be built gradually outwards until all the BISs have been modified.

f) When all the BISs in the confederation have been modified, the managers of remote BISs can be informed that the confederation has been fully formed, and any old policies regarding the RDs in the confederation can now be safely deleted.

Note that the above rules apply as well if the new confederation is one that is nested within another confederation. The only difference that occurs when the new confederation to be formed is nested within a confederation X is that managers of BISs that are not contained within X do not need to be informed about the formation of X.

Also note that the BISs internal to X still need to retain their policies regarding the RDs and confederations within X.

I.2 Forming a Higher Level Confederation

Now assume it is desired to form a new confederation X that will have some number of already formed confederations nested within it, say Y and Z. The steps are:

a) As above, warn all managers of all affected BISs (i.e. in the lowest level confederation(s) that X will be nested within (or all BISs, if X is a top level confederation)) that a new confederation X will be formed, and list all the RDs and confederations to be included (the ones that are currently visible externally, i.e., don't list the RDs in confederations to be included in X -- just list the top level confederations to be included)..

b) As above, managers of BISs must figure out a reasonable policy for the new confederation.

c) As above, the policy regarding the new confederation must be ADDED to the existing set of policies (the confederation will not appear immediately).

d) As above, when ample time has elapsed so that managers will have been given an appropriate opportunity to modify the policies at their BISs, the managers of the BISs in the new confederation can start informing the BISs in the confederation about the confederation.

e) As above, one by one, each BIS is informed that it is in the confederation, where the order in which the BISs are configured with the confederation information is critical—at all times the confederation must be connected.

The difference, though is in how the BISs are configured. Initially, the BISs are informed, one by one, that they are in X, but they are NOT informed that Y and Z are nested within X. Instead, they will be configured as though X is a lowest level confederation.

f) After all BISs in the confederation have been configured to know they belong to X, they can one by one be modified to believe Y and Z are nested within X. In contrast to the knowledge that they belong to X, which must be configured in a careful order, the knowledge that Y and Z are nested within X can be configured within the BISs in X in any order.

g) When all the BISs in the confederation have been twice modified (once to know about X, and once to know about the nesting rules), managers of remote BISs can be informed that the confederation has been fully formed, and the policies regarding RDs and confederations in the new confederation can now be safely deleted.

I.3 Deleting a Lowest Level Confederation

Now suppose there is a confederation X, with no confederations nested within it, that is being dissolved. The steps involved are:

a) First warn all managers of all affected BISs (see point 1 in the previous 2 sections for a rigorous description of which BISs are affected), that X will be dissolved, and list all the RDs in X.

b) A manager of a BIS that has policies regarding X needs to add the same policy many times, one for each RD in X. It is also possible at this time to make policies that are different for the RDs in X.

c) When ample time has elapsed so that managers will have been given an appropriate opportunity to modify the policies at their BISs, the managers of the BISs in X can start informing the BISs in X to forget about X.

d) One by one, each BIS in X is informed that it is not in X. The order in which the BISs are modified is critical. At all times the set of BISs that believe they are in X must be a connected set. Thus X must be shrunk gradually towards one point.

e) When all the BISs in X have been modified, the managers of remote BISs (those in the confederation within which X had been nested, or all BISs if X was a top level confederation) can be informed that X no longer exists, and the policies regarding X can now be safely deleted.

I.4 Deleting a Higher Level Confederation

The steps involved are:

a) As above, warn all managers of all affected BISs (see point 1 in the previous 3 sections) that confederation X will be dissolved, and list all the RDs and confederations included in X (i.e., the ones that will become visible when X is deleted.)

b) As above, policies need to be ADDED regarding all the RDs and confederations that were included in X.

c) As above, when ample time has elapsed so that managers will have been given an appropriate opportunity to modify the policies at their BISs, the managers of the BISs in the new confederation can start informing the BISs in X about X′s impending dissolution.

d) Now different from above, one by one (in any order) the BISs in X are informed that nothing is nested within X any more, though they retain knowledge of X.

e) After all BISs in X have been configured to believe X is a bottom level confederation, knowledge of X can be carefully deleted from the BISs (careful because the order is critical, as above, i.e. X must at all times be connected.)

f) After all BISs previously in X have been twice modified (once to delete the nesting rules for X, and one to delete X), managers of remote BISs can be informed that X has been fully dissolved, and policies regarding the confederation can now be safely deleted.

Annex J. Example Usage of MULTI-EXIT_DISC Attribute

(Informative)

The MULTI-EXIT DISC attribute can be used to provide a limited form of multi-path (load-splitting), as is shown in the following examples.

— Example 1 (see Figure 12):

Consider the case when a BIS A located in routeing domain RD-A has two adjacent BISs (B1 and B2) that belong to the routeing domain RD-B. Assume that RD-B has Network Layer Reachability information about NSAPs N1, N2, ... Nk, and it wants to advertise this information to RD-A. By using the MULTI-EXIT_DISC attribute RD-B may do selective load splitting (based on NSAP addresses) between B1 and B2.

For example, BIS B1 advertises to BIS A Network Layer Reachability information N1, N2, ... Nm with

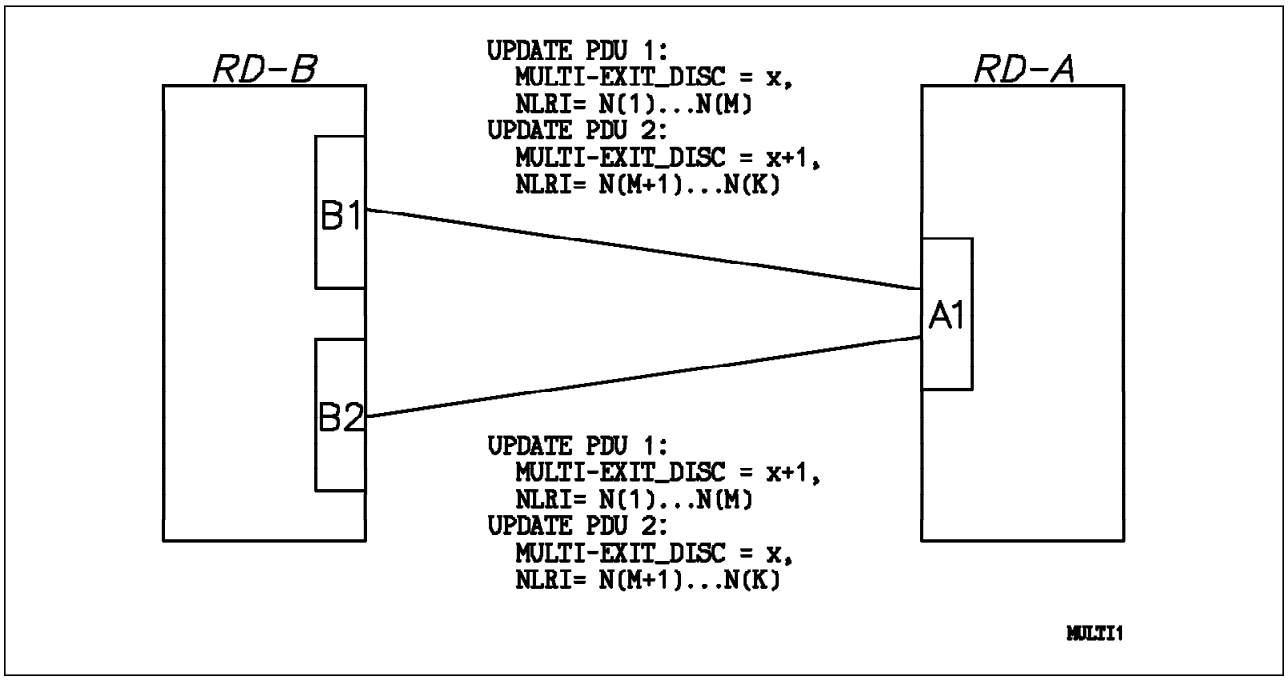

Figure 12. Example 1 Configuration

the MULTI_EXIT_DISC set to X, and advertises N(m+1), ... Nk with the MULTI_EXIT_DISC set to X + 1.

Similarly, BIS B2 advertises to BIS A Network Layer Reachability information N1, N2, ... Nm with the MULTI_EXIT_DISC set to X + 1, and advertises N(m+1), ... Nk with the MULTI_EXIT_DISC set to X.

As a result, traffic from BIS A that is destined to N1, N2, ... Nm will flow through BIS B1, while traffic from BIS A that is destined to N(m+1), ... Nk will flow through BIS B2. This scenario illustrates the simplest way of doing limited multipath with IDRP.

— Example 2 (see Figure 13):

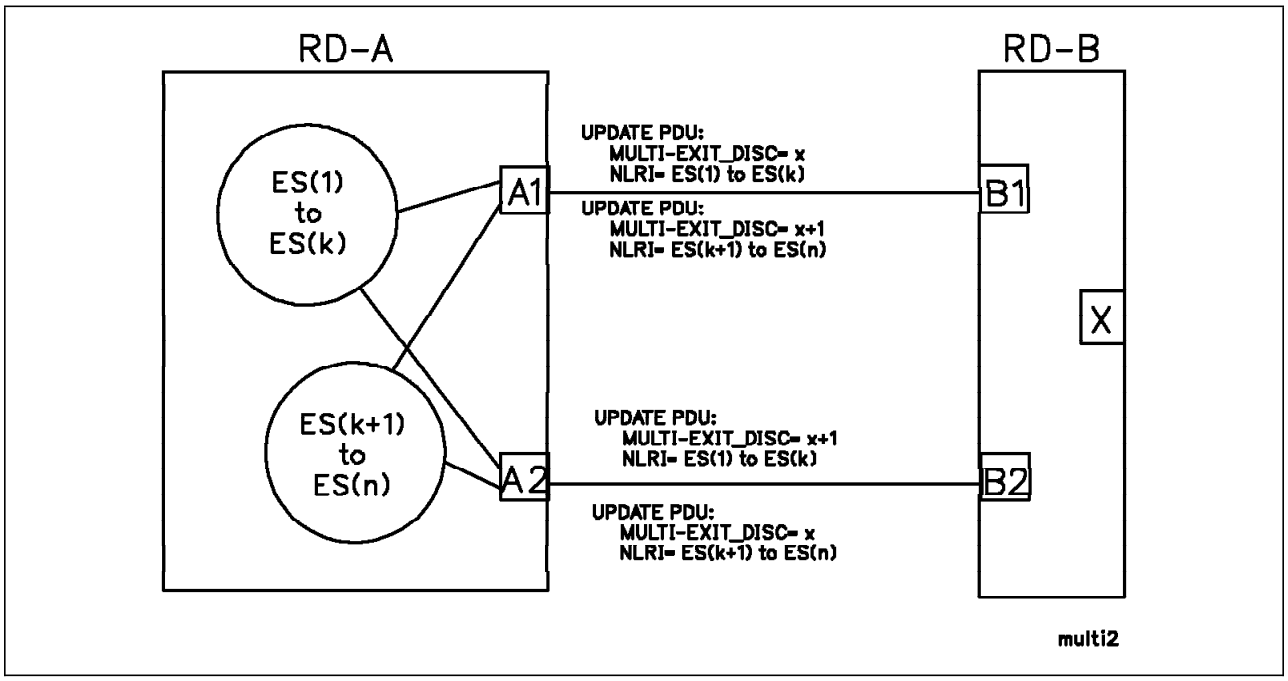

Figure 13. Example 2 Configuration

Next consider more complex case where there is a multihomed routeing domain RD-A that has only slow speed links. RD-A is connected at several points to a transit routeing domain RD-B that has only high speed links; BIS A1 is adjacent to BIS B1, and BIS A2 is adjacent to BIS B2. RD-A wants to minimize the distance that incoming NPDUs addressed to certain ESs—say ES(1) through ES(k)—will have to travel within RD-A.

One way of doing this is by making BIS A1 to announce to BIS B1 destinations ES(1) – ES(k) with a lower MULTI_EXIT_DISC, as compared to the MULTI_EXIT_DISC that BIS A2 will use when announcing the same destinations to the BIS B2.

Similarly, BIS A2 would announce to BIS B2 destinations ES(k+1) – ES(n) within the RD-A that are closer to the BIS A2 (than to the BIS A1) with the lower MULTI_EXIT_DISC, as compared to the MULTI_EXIT_DISC that the BIS A1 will use when announcing the same destinations to the BIS B1.

When traffic that is destined to some ES within RD-A enters RD-B on its way to RD-A via BIS X, X picks up the exit BIS that has the lowest MULTI_EXIT_DISC value for that destination. For example, X may pick up BIS A2 as an exit, even if the distance between A2 and X is greater than the distance between A1 and X.

Annex K. Syntax and Semantics for Policy

(Informative)

This annex describes an example of a policy syntax and its associated semantics for the protocol defined in this international standard. The example is intended to be informative: that is, alternative syntaxes with equivalent richness of functionality are not precluded, and other mechanisms may be needed to provide a fully functional configuration language.

K.1 Overview

The policy information base allows routing domain administrators to control routing information usage and flow according to the policies of the domain. The policy information base is made up of three component sections, corresponding to three primary types of policy concerns that have been identified:

a) *Route preference* assigns a preference value to incoming routes; this is the ″local selection policy″ regarding routes. These policies determine which routes in the Adj-RIBs-In are selected for the LOC-RIB.

b) *Route aggregation* chooses routes for aggregation and expresses some control over how aggregation is performed. These policies select routes in the LOC-RIB that are to be advertised as an aggregate. These policies can affect routes sent to BISs internal and external to the domain.

c) *Route distribution* modifies and selects routes for redistribution; this expresses the domain′s ″transit policy″. These policies control traffic through the domain by restricting which routes from the LOC-RIB are placed in the RIB-OUTs. Modifications may affect routes sent to internal or external BISs, however, selection policy only affects the distribution of routes to BISs external to the domain; internal BIS neighbors receive route information from the local BIS regardless of policy.

Each policy subsection is comprised of a list of *policy statements* that express the domain′s policy. Although the policy statements of each section are different, all include a *route pattern* (which is a template for matching route attributes) and the associated *actions*. A domain administrator can use these ″match + action″ pairs to express the administrative policy of the routing domain.

K.1.1 Preference Statement

The preference statement is identified by the ″PREF″ keyword, and has the following format:

```
PREF <route pattern template> [<local_cond>] [<bis>]
  = <preference value expression>
```

A PREF statement assigns a value to any route (from a BIS neighbor in an external domain) that matches the specified pattern. The assigned value determines the degree of preference that will be used in the Decision Process. This value is also used to generate the LOC_PREF attribute. Note that it is possible for the assigned value to be less than zero or greater than 255. Conversion from the assigned value to an eight-bit LOC_PREF field is a local matter. Routes received from internal BIS neighbors will already have a LOC_PREF field. The use of the LOC_PREF field as a basis for selecting the most preferred route is described in clause 17.12.8.

The components of a <route pattern template> are:

— <nlri>
— [<info_src>]
— [<path>]
— [<dist_att>]
— [<att_cond>], where:.

<nlri> : Reachable destinations; matches if the actual route′s NLRI is a subset of the destinations specified by this template. The <nlri> must be present in the route pattern of every policy statement.

<info_src> : Can be ″idrp″|″ext″|″info_any″, which is matched base d on the presence/absence of the EXT_INFO attribute in a route. These tokens are optional; if not present, the default match is ″idrp″.

<path> : Regular expression over RDIs to match against the content of the RD_PATH attribute. A <path> is optional; if not present, the default matches any RD_PATH attribute.

<dist_att> : Specifies a set of distinguished attributes for a route match. The <dist_att> is optional; if not present, the pattern matches routes with any set of distinguished attributes.

<att_cond> : Provides matching/control for all other attributes, i.e. other than what is carried in RD_PATH, EXT_INFO path attribute, and the presence of distinguished attributes. This specifies conditions of route attributes that must be met for a route to match, e.g. (EXPENSE() < 10) && (! present(DIST_INCL)) might be the condition if the intent is to match a low-cost route which does not have certain re-distribution restrictions. No <att_cond> need be specified; if not present, the route pattern matches routes with any attributes.

Note that the route pattern template is found in all three types of policy statement (preference, aggregation, and distribution). A slightly different form is used in the aggregation policy statement, which is discussed below.

The PREF statement (actually, all policy statements) may also include ″local condition tests″, which allow policy to be sensitive to criteria not related to a route′s attributes (e.g. time of day). A <local_cond> is optional; if not present, routes are matched under any local conditions.

The specification of <bis> allows routes from different BIS neighbors to be assigned preferences differently. Any number of external BIS neighbors may be specified, and only routes received from these neighbors will be assigned a preference value by the statement.

The <preference value expression> is an integer arithmetic expression with operators ′+′, ′-′, ′*′, ′/′, and (similar to the C language) a conditional operator ′?′. The basic operands are constants, or pre-defined functions which return values based on the attributes of a route, e.g. hopcount(), capacity(), weighted_list(EXCL,<table>). The condition expression for the condition operator includes the logic operators ″&&″ and ″||″, and may include (1) tests for the presence of an attribute, (2) comparisons of integer expressions including attribute values, and (3) local condition tests. A <pref value> is required in all preference statements.

The order of PREF statements in a configuration file is significant; the first <route pattern> that matches an incoming route will assign the preference value. The list of PREF statements can be thought of as filters, each acting on particular routes; a routing domain administrator can make effective use of this first-match functionality by listing more specific route patterns early and more general patterns later. Hence, the ″filters″ start at a fine degree of granularity to assign preference to routes of particular importance, while other routes are handled by increasingly general ″filters″.

If a route does not match any <route pattern>, it is dropped and not considered by the IDRP Decision Process. Note that there may be over-riding operational criteria that dictate that the non-matched routes can not be handled in this manner.

The concept of decreasingly specific filters is useful for all of the policy sections: preference, aggregation, and distribution. As described below, more flexible control of the processing sequence for aggregation and distribution statements is possible, and necessary to concisely express policy.

K.1.2 Aggregation Statement

The aggregation statement is identified by the ″AGGR″ keyword, and has the following format:

```
AGGR <route pattern> <local_cond> =
 [<recipient BIS>] <aggr_nlri> [″DONE″|″CONT″]
```

The <route pattern> specification of the aggregation statement is slightly different than the PREF statement <route pattern>. The only difference is that the <nlri> template will consist of two NLRI specifications separated by the ″MUST″ token, i.e. <nlri> ″MUST″ <nlri>. The first <nlri> is used to match routes that can be aggregated, while the second <nlri> specifies NLRI which must be present for the aggregate route to be instantiated. Either of the <nlri> specifications may omitted, but not both. If the second <nlri> is omitted, the ″MUST″ token is not required.

The AGGR statement′s <local conditions> template has the same syntax and semantics as the PREF statement.

The <recipient BIS> of the aggregation statement indicates which external BIS's RIB-OUTs are to receive the results of the statement's route aggregation. One, several, or all external BISs may be specified to receive the aggregate route "manufactured" by an AGGR statement. In addition, an administrator can specify "internal_bis" to affect aggregation to all other BISs internal to the routing domain.

If a BIS is included in the recipient list, it will receive the aggregated route but not the component routes; if an aggregate is not instantiated to a particular BIS, it will receive all of the component routes. Note that by using additional AGGR statements (with more specific route matching templates), particular component routes may be advertised separately from the aggregate route. If the <recipient BIS> list is not specified, the default action is to announce the aggregate route to all external neighbor BISs; the default action will announce component routes to internal BISs.

The <aggr_nlri> specifies how the BIS determines which NLRI to advertise for the aggregate route. The two primary specifications are manual ("man") or automatic ("auto"), with two additional tokens ("auto_short" and "auto_subset") to specify variations of "auto"; "auto" includes both of these variations. Automatically aggregated NLRI will only reduce routes if there is no loss of reachability information, i.e. it will only advertise a more general NLRI if it can algorithmically determine that the aggregate is not advertising NLRI other than those of the component routes. Domain administrators can also "manually" override the automatic aggregation and specify that aggregated route NLRI may include destinations not included in any component of the aggregate route. The "manual" option is primarily intended for use when additional (complete) information is known about the NLRI (e.g. when it is part of the address space under control of the routing domain). It is assumed such information is obtained by means outside of IDRP. For instance, using "manual" NLRI configuration, a domain that acts as an address assignment authority may announce a single prefix for all routes containing longer extensions of this prefix, even though portions of the address space may be unassigned, with no route available to some destinations advertised by the NLRI. Manually aggregated NLRI is determined by taking the longest common prefix of the set of NLRI specified by the route pattern <nlri>. Using automatic aggregation, the aggregate NLRI is computed to be the shortest NLRI prefix necessary to announce the component route's NLRI (the aggregate NLRI is also the longest common prefix of the component routes). The two variations of "auto" are as follows: (1) "auto_short" will collapse several longer NLRI prefixes into a single common prefix based on the binary representation, e.g. XX:YY:0xF601:* - XX:YY:0xF60F:* will be advertised as XX:YY:0xF60:*, and (2) "auto_subset" will permit longer prefixes to be aggregated with shorter ones, e.g. XX:YY:ZZ:* would be aggregated with XX:YY:* into XX:YY:*.

Like PREF statements, the AGGR statements are applied in sequence (they are applied to the set of routes in the LOC-RIB). "DONE" and "CONT" provide control over additional processing of routes by subsequent AGGR statements. "CONT" is used to indicate that the aggregate route may be treated as a component route by later AGGR statements, and thus may be matched and further aggregated. "DONE" indicates that the aggregate is to be advertised as-is, and will not be considered as a component route for further aggregation. Specification of "DONE" or "CONT" is optional; the default case is "DONE".

[Note: Aggregated routes will have a preference value assigned by the policy PREF statements; just as incoming routes from other BISs, aggregated routes are processed by the route preference statements. If an aggregate route does not match a PREF statement template, no value is assigned and the aggregate is not instantiated.]

K.1.3 Distribution Statement

The distribution statement is identified by the "DIST" keyword, and has the following format:

```
DIST <route pattern> [<local_cond>] = [<recipient BIS>]
  <select_action> [<modifications>] ["DONE"|"CONT]"]
```

The <route pattern> for the DIST statement is the same as the PREF statement <route pattern>, and <local_cond> serves the same function for the DIST statement as it does for the AGGR and PREF statements.

Similar to the AGGR statement, the <recipient BIS> specifies which RIB-OUTs are effected by the statement. The RIB-OUTs associated with the neighbors specified in <recipient BIS> may be affected in three ways by a DIST statement: (1) the route may be modified in these RIB-OUTs, (2) the route may be placed in, or removed from, the RIB-OUTs, and (3) the route may be marked as "DONE", so that it remains unaffected by further DIST statements.

The <select_action> can be ″select_on″, ″select_off″, ″select_only″, or ″modify″; these control whether a route is distributed to an adjacent BIS. If a route is selected for advertisement to a particular BIS neighbor, it will be placed in the associated RIB-OUT. By default, routes are not selected for advertisement until selected by a DIST statement. The semantics of the <select_action> effect this distribution as follows:

″**select_on**″ The route should be placed in RIB-OUTs associated with all specified neighbors, unless ″selected off″ by later DIST statement.

″**select_off**″ The route should not be placed in RIB-OUTs associated with the specified neighbors, unless ″selected on″ by a later DIST statement.

″**select_only**″ The route should be placed in RIB-OUTs associated with all specified neighbors, unless ″selected off″ by later DIST stmt; in addition, the route should not be placed in RIB-OUTs associated with BISs not in <recipient BIS>, unless ″selected on″ by a later DIST statement.

″**modify**″ Modify only; the selection status of routes are not effected by this DIST statement. Presumably, some routes matching this statement will also match, and be selected for distribution by, other DIST statements.

The effects of a <select_action> is applied only when <recipient BIS> indicates a BIS in an adjacent domain. It has no effect on distribution to BISs within the same domain as the local system.

Note that in most cases, only the routes in RIB-OUTs specified by <recipient BIS> will be affected by a DIST statement, however, there is one exception. The ″select_only″ action also indicates routes are not to appear in the RIB-OUTs associated with BISs not in <recipient BIS> list, and that these routes may or may not be considered for further DIST statement processing (in the excluded BISs) based on the DIST statement′s DONE/CONT token. Using ″select_only″ along with ″DONE″ allows one to concisely specify that only certain BISs are to receive particular routes, and as an additional effect, make certain these routes are not inadvertently selected for other BISs by a subsequent DIST statement that matches a more general route pattern.

A list of <modifications> statements indicates policy-driven changes to route attributes (e.g. DIST_LISTs, HIERARCHICAL RECORDING changes, etc). No <modifications> need be present; the default leaves routes unchanged.

″CONT″ and ″DONE″ have similar function as in the aggregation statement; they control whether routes matching a particular DIST statement may be affected by later DIST statements. ″CONT″ indicates that a matched route in a specified RIB-OUT is eligible for further modifications, ″DONE″ indicates no further DIST statement processing. Specification of ″DONE″ or ″CONT″ is optional; the default case is ″DONE″.

K.2 Policy Configuration Language BNF

This section specifies the basic syntax for this example IDRP configuration language. This BNF tree does not include all terminal-symbol leaves; it is sufficient as an illustration of some minimal useful functionality, however, it is not complete.

The policy configuration language uses a ′#′ to denote a comment to the end of line. This convention is also used to provide comments throughout the BNF specification. This BNF uses square brackets, ′[′ and ′]′, as a notational convenience to indicate optional (zero or one occurrence) syntactic symbols. This BNF also uses curly braces, ′{′ and ′}′ and a ′|′ to indicate a choice of symbols.

A discussion of the semantics of this language can be found in K.1.1 above..

K.2.1 PREF Statement BNF

```
<preference_section> ::= <p_stmt_list>
<p_stmt_list> ::= <p_stmt> ';' <p_stmt_list> | <empty>
<p_stmt> ::= "PREF" <nlri> <route_pattern> <local_cond> [<bis>]
      '=' <preference_value_expression>
```

All of the symbols used by the <p_stmt> are also used in other places, and are defined in K.2.4.

K.2.2 AGGR Statement BNF

```
<aggregation_section> ::= <a_stmt_list>
<a_stmt_list> ::= <a_stmt> ';' <a_stmt_list> | <empty>
<a_stmt> ::= "AGGR" <nlri_2> <route_pattern> <local_cond>
'=' [<bis>] <aggr_nlri> <done_cont>

<nlri_2>::= '{' <dest_list> [ "MUST" <dest_list> '}' ]
<aggr_nlri>::= "auto" | "auto_subset" | "auto_short" | "man"
```

K.2.3 DIST Statement BNF

```
<distribution_section> ::= <d_stmt_list>
<d_stmt_list> ::= <d_stmt> ';' <d_stmt_list> | <empty>
<d_stmt> ::= "DIST" <nlri> <route_pattern> <local_cond>
      '=' [<bis>] <select_action> <mods> <done_cont>
<select_action> ::= "select_on" | "select_off" |
"select_only" | "modify"
```

Policy- defined changes to route attributes are distinct from attribute updates that occur due to basic "operational" processing (e.g. HOP_COUNT is updated without regard to policy).

```
<mods> ::= '{' <mod_list> '}' | <empty>
<mod_list> ::= <mod_statement> ';' <mod_list> | <empty>
<mod_statement> ::= <multi_exit_statement> |
    <dist_list_statement> |
    <hrecord_statement>  |
    <next_hop_statement>

<multi_exit_statement> ::= "set_multi_exit" '(' <value> ')'
```

init_hr() - If HIERARCHICAL_RECORDING attribute is not already present in route, add attribute to route and initialize to one (1) to limit distribution within RDC.

```
<hrecord_statement>  ::=  "init_hr" '(' ')'
```

Add RDIs to INCL or EXCL list.

```
<dist_list_statement> ::=
"allow_dist" '(' <rdis> ')' |
"prohibit_dist" '(' <rdis> ')'

<next_hop_statement> ::=
"set_next_hop" '(' <net> <snpa> ')'
```

K.2.4 Common BNF Symbols

This section describes common syntax components used by all three types of policy statements.

K.2.4.1 Route Attribute Matching Template

Reachability:

```
<nlri> ::= '{' <dest_list> '}'
<dest_list> ::= <dest> ',' <dest_list> | <dest>
<dest> ::=    "nlri_any"          |
  ["not"] <nsap> ':' '*' |    ## prefix match
  ["not"] <nsap>       |    ## exact <nsap> match
  <empty>
```

Route matching template

```
<route_pattern> ::= <info_src> <path> <dist_att> <attrib_cond>
<info_src> ::= "idrp" | "ext" | "info_any"
<path> ::= '/' <<regular-expression over RDIs>> '/'
```

Distinguished attributes

```
<dist_att> ::= <empty> |
"dist_att_none" |
"dist_att_any" |
'(' <qos> <security> <priority> ')'
(If <empty>, default is "dist_att_any".)

<qos> ::= "qos_any" | <qos_list>
<qos_list> ::= <one_qos> <qos_list> | <empty>
(If <empty>, default is "qos_any".)

<one_qos> ::= "qos_none" | "error" | "expense" | "delay" |
"capacity" | <src_qos> | <dst_qos>
<src_qos>   ::= "srcqos"  <nsap>  <qos_value>
<dst_qos>   ::= "dstqos"  <nsap>  <qos_value>
<qos_value> ::=  ## TO BE DEFINED ##

<security> ::= "security_any" | <sec_list> <sec_list> ::= <sec_list> <one_sec> | <empty>
(If <empty>, default is "security_any".)

<one_sec> :: = "security_none" | <srcsec> | <dstsec>
<srcsec> ::= "srcsec" <nsap>
<dstsec> ::= "dstsec" <nsap>
```

Security-related BNF is subject to change as the protocol continues to develop.

```
<priority> ::= "priority_any" | "priority" | "priority_none"| <empty>
```

The "priority_any" token matches routes in either case, whether; thef priority attribute is present, or if it is not. If <empty>, default is "priority_any".

K.2.4.2 BNF: Numerical Expressions

```
<value> ::=
<integer>     |
<att_value>    |
'('<value> ')' |
<value> <integer_op> <value> |
<case_statement>
<case_statement> ::=  '(' <case_cond> '?' <value> ':'<value> ')'
<att_value> ::=
"hopcount"   "()"            | ## rd_hopcount
"pathweight" '(' <table> ')'     | ## weighted path
"listlen"   '(' {"INCL"|"EXCL"} ')'            |
"listweight" '(' {"INCL|"EXCL"} ',' <table> ')'     |
<att_value_name> "()"
```

Returns the value carried by the attribute specified by the <att_value_name>.

```
<att_value_name> ::= "multi_exit" | "loc_pref" | "priority"
| "delay" | "expense"  | "error" | "capacity"
| "hier_rec"
```

This example of the PIB BNF does not deal with the following attributes: SRC_QOS, DST_QOS, SRC_SECURITY, and DST_SECURITY.

K.2.4.3 BNF: Conditional Specification

There are three related types of conditions:

a) <attrib_cond> used when doing a route match; only tests/examines attributes of a route
b) <local_cond> used in policy actions; tests "other" (TBD) criteria (e.g. time of day)
c) <case_cond> used in case statement; may test attribute or local criteria

```
<local_cond> ::=
 <A_cond_LOCAL>              |
 '!' <local_cond>            |
 '(' <local_cond> ')'        |
 <local_cond> "&&" <local_cond>  |
 <local_cond> "||" <local_cond>

 <A_cond_LOCAL>  ::=  ## TO BE DEFINED
```

This is currently a place holder reserved for future use; one potential example is time of day.

```
 <attrib_cond> ::=
 <A_cond_ATTRIB>             |
 '!' <attrib_cond>           |
 '(' <attrib_cond> ')'       |
 <attrib_cond> "&&" <attrib_cond> |
 <attrib_cond> "||" <attrib_cond>

 <A_cond_ATTRIB> ::= <att_value> <compare_op> <value>    |
 "present" '(' <attribute_name> ')'  |
 <other_att_test>

 <case_cond> ::=   <A_cond_CASE>          |
 '!' <case_cond>             |
 '(' <case_cond> ')'         |
 <case_cond> "&&" <case_cond>  |
 <case_cond> "||" <case_cond>

 <A_cond_CASE>  ::= <A_cond_ATTRIB> | <A_cond_LOCAL>

 <attribute_name> ::= "src_qos" | "dst_qos" |
 "dst_sec" | "src_sec" |
 "dist_incl" | "dist_excl" |
 "ext_info" | "next_hop" |
 <att_value_name>

 <other_att_test> ::= <next_hop_test>
```

This PIB BNF only defines the next_hop_test; others may be defined.

```
 <next_hop_test> ::= "next_hop" '(' <next_hop_list> ')'
 <next_hop_list> ::= <next_hop_match> ',' <next_hop_list>   |
 <next_hop_match>
 <next_hop_match> ::= "next_hop_any"         |
 ["not"] <net> ':' '*'     |
 ["not"] <net> [<snpa>]    |
 ["not"] <net> <one_snpa>  |
```

One can attempt to match NEXT_HOP attribute against "any", a set of BISs (NET prefix), a particular BIS and optionally specific interfaces. Also one can match routes against local interface over which route was received.

K.2.4.4 Other Common BNF Symbols

<bis> ::= "bis_all" | '{' <bis_list> '}' <bis_list> ::= <bis_item> ',' <bis_list> | <bis_item> <bis_item>
::= "rdi" <one_rdi> | "bis" <net> |
"internal_bis" | "external_bis"

One can specify all BIS neighbors in an adjacent RDIrdi, single out a particular bis by NET, specify all internal BIS neighbors, or all external BIS neighbors.

<done_cont> ::= "DONE" | "CONT" | <empty>
(Default <empty> is DONE)

<table> ::= '{' <table_list> <table_default> '}'
<table_list> ::= <table_pair> <table_list> | <empty>
<table_pair> ::= '(' <one_rdi> ',' <integer> ')'
<table_default> ::= '(' "default" ',' <integer> ')' | <empty> p.List of interfaces of this BIS;

<snpa> ::= '{' <snpa_list> '}'
<snpa_list> ::= <one_snpa> ',' <snpa_list> | <one_snpa>

Routing Domain Identifiers;

<rdis> ::= '{' <rdi_list> '}'
<rdi_list> ::= <rdi_list> ',' <one_rdi> | <one_rdi>

K.3 Simple Example

This example is provided to make the intended use of the policy configuration language more clear. Note that this example is incomplete, and at best only marginally realistic; it is intended to illustrate the basics of the policy configuration statements for purposes of this overview.

Throughout this text we refer to the set of distinguished attributes which has no QOS attribute, no priority attribute, and no security attribute as the default set of distinguished attributes. This is the distinguished attribute set specified by "dist_att_none".

Consider the portion of an internet shown in Figure 14. Assume that each routing domain has exactly one BIS that communicates with all adjacent domains' BISs.

K.3.1 Transit Domain 3

Example policies of transit domain RD #3 might be as follows:

a) RD #3 only accepts IDRP originated routes. It supports two sets of distinguished RIB_ATTs: the default set (no distinguished attributes) and the set having only the CAPACITY QOS attribute.

b) Routes with CAPACITY QOS must travel via RD#6; CAPACITY must be greater than 15.

c) For routes with no distinguished attributes, prefer routes through transit domain 1, however preference should be given to routes with short paths; large hop counts on a route via RD#1 may cause a shift to another transit domain.

d) CAPACITY QOS routes are only offered to some domains (RDs #5, #8), and are restricted from being propagated further (i.e. via the DIST_LIST_INCL attribute).

e) All routes with no distinguished attributes are re-distributed to every neighbor RD; hierarchical recording is desired to limit distribution of all of these routes (the specific RDC membership information is irrelevant for this example).

f) Any route (default or CAPACITY) which pass through transit RD#9 (not pictured) can only be redistributed to some domains (RD#2, RD#5, RD#8).

g) All routes carrying NLRI of the address space controlled by domain #3 (XX:YY:3:*) will be aggregated (regardles s whether or not aggregated routes include NLRI for all of this space). In addition, routes car-

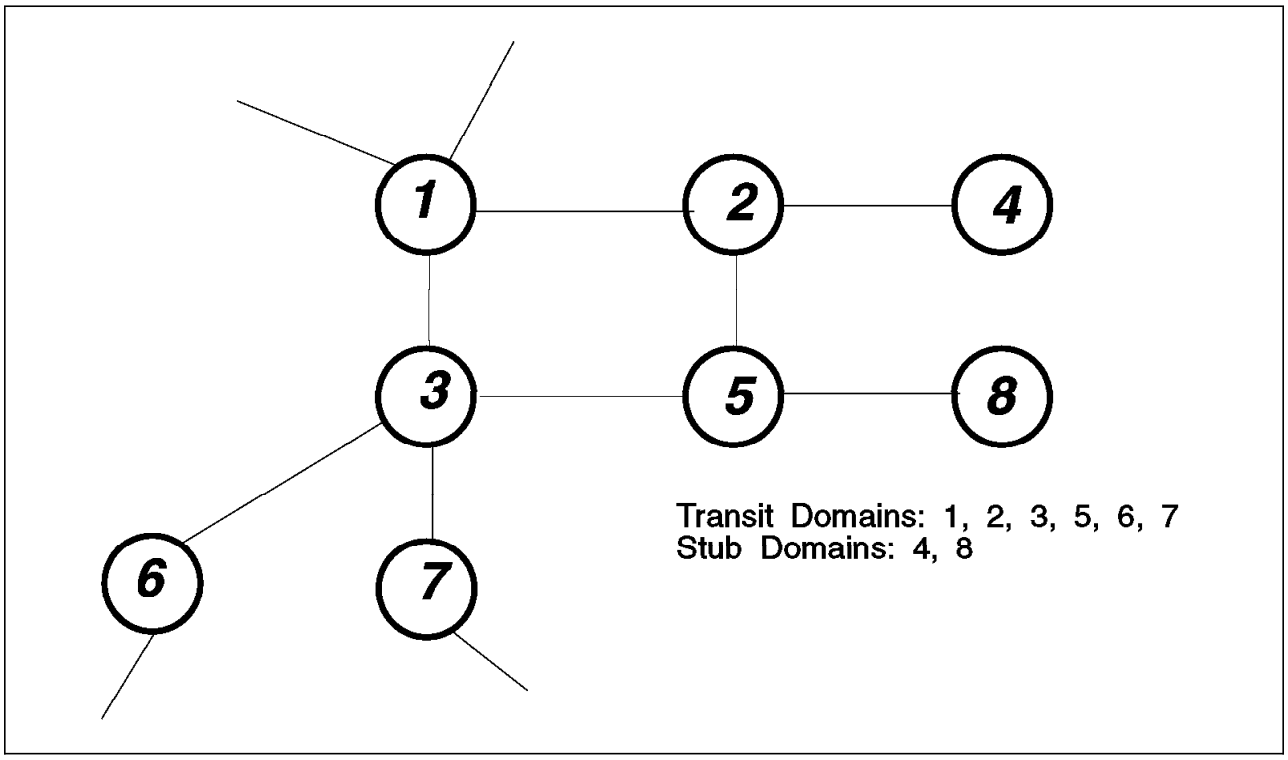

Figure 14. A Portion of an Internet

rying NLRI for RD#5 NSAPs (XX:YY:3 :5:*) will be announced separately. All routes for default dist_atts will be aggregated algorithmically. A "default route" (zero length NSAP) will be advertised for CAPACITY QOS routes (although distribution of this route will be limited to particular domains, i.e. RD#5, by the select/modify policy section).

K.3.2 Policy Configuration Example

The following is one example of an expression of the above policies using this configuration language. The next subsection, examines and discusses each line of this configuration example in detail.

This example assumes that the policy language is case insensitive.

```
PREF {nlri_any} / .* 6 / (CAPACITY security_none priority_none)
(CAPACITY() > 15) = 50;
PREF {nlri_any} / .* 1 / dist_att_none = 255 - hopcount();
PREF {nlri_any} /.*/    dist_att_none = 245 - hopcount();
AGGR {XX:YY:3:5:*} /.*/ dist_att_none = man;
AGGR {XX:YY:3:*}  /.*/ dist_att_none = man;
AGGR {nlri_any}   /.*/ dist_att_none = auto;
AGGR {nlri_any}  /.*/ (CAPACITY security_none priority_none) = man;
DIST {nlri_any}  /.* 9 .*/ =
modify {allow_dist({RD#2, RD#5, RD#8});} CONT;
DIST {nlri_any}  /.* 6/ (CAPACITY security_none priority_none)
CAPACITY() > 15) = {BIS#5} select_only {allow_dist(RD#5, RD#8);};
DIST {nlri_any} /.*/  = select_on {init_hr();};
```

K.3.3 Discussion

Each policy statement given in section 4.2 is discussed. In most cases, the optional parts of the BNF have been omitted if the default action is appropriate to represent the example policy. For brevity, these defaults will be mentioned in the discussion only once, at the statement where they are first encountered.

K.3.3.1 Preference Statement Discussion

Recall that the sequence of statements is significant for determining the application and processing of all types of policy statements. Preference statements are the least complex of the three; routes are simply assigned the preference associated with the first <route pattern> that is matched (the other statements′ processing and application sequence are discussed later in this text).

The first PREF statement matches routes to any destination, indicated by {nlri_any}. No token for information source is present, so by default only routes where the information source was IDRP are matched (i.e. routes where no EXT_INFO attribute is present). The third expression ″/ .* 6 /″ matches any RD_PATH attribute where the last ″hop″ was from RD#6.

```
PREF {nlri_any} / .* 6 / (CAPACITY security_none priority_none)
 (CAPACITY() > 15) = 50;
```

The 3-tuple (CAPACITY security_none priority_none) indicates a route is to match if it corresponds to the RIB_ATT which has CAPACITY QOS, no security, and no priority. Finally, the attribute conditions only allow routes with CAPACITY attribute greater than 15 to be matched. There are no local conditions to be considered, so nothing is specified and by default routes are matched under any local conditions. Note that none of the actions in this example are dependent on local conditions, so this will be ignored for the rest of the example. Routes matched by this pattern are simply assigned a preference of 50.

The second PREF statement also matches routes to any destination, if the routing information source was IDRP. The statement matches routes received directly from RD#1, by examining the route′s RD_PATH attribute. The ″dist_att_none″ is specified, so only routes which have no QOS attribute, no priority attribute, and no security attribute will be matched. There are no other attribute or local conditions to meet, which is the default if nothing is specified.

```
PREF {nlri_any} / .* 1 / dist_att_none = 255 - hopcount();
```

This statement assigns a route preference based on the HOP_COUNT value plus a constant. The constant (255) is relatively ″good″ (relative to 245 in the next statement) so that routes through RD#1 are preferred (per policy C above). Since routes will be assigned preference by the the first <route pattern> matched, a path through RD#1 matching this pattern will not have a value assigned by the next statement, even though it has a more general <route pattern> and also would be a correct match.

The next PREF statement matches any route with no distinguished attributes, again, only if the information source was IDRP.

```
PREF {nlri_any} /.*/    dist_att_none = 245 - hopcount();
```

Routes matching this pattern are assigned a preference based on the hop count and a relatively ″bad″ constant (245), so that these routes are preferred less than routes through RD#1 (which match a previous PREF statement).

Examining the last two preference statements, per policy C routes through RD#1 are preferred unless the path length (hop count) is worse (by ten hops or more).

K.3.3.2 Aggregation Statement Discussion

Aggregation statements are also processed in the order that they appear, however, their processing and application is not simply based on first match. An AGGR statement may be marked with ″CONT″ to indicate that the aggregate route may act as a component for subsequent AGGR statements. Alternatively, ″DONE″ indicates that an aggregate should be installed/advertised, and not considered in further aggregation processing.

The first aggregation statement matches routes with a specific set of destinations, {XX:YY:3:5:*}, and announces the aggregate route with manually configured NLRI. The longest common NLRI prefix specified is XX:YY:3:5, so this will serve as the NLRI for the aggregate route. Using ″manual″ aggregation, this aggregate is instantiated whether or not the NLRI of the matched component routes include all destinations implied by the prefix XX:YY:3:5.

```
AGGR {XX:YY:3:5:*} /.*/ dist_att_none = man;
```

Any number of routes may match this pattern, and are replaced by a single aggregate route. No recipient <bis> are specified, so by default the aggregate route (rather than the components) is announced to all external BIS neighbors. The default action, ″DONE″, requires that the aggregate route be distributed without undergoing further aggregation. Hence, routes to these destination NSAPs are announced separately from the rest of the XX:YY:3:* NSAPs (which are aggregated below).

The second AGGR statement is almost identical to the first.

```
AGGR {XX:YY:3:*} /.*/ dist_att_none = man;
```

Routes to a specific set of NSAPs are matched and aggregated; these destinations are a superset of those matched by the previous AGGR statement. This construct (two AGGR statements with overlapping NLRI) can be used to make certain that particular longer prefixes are announced separately from a more general aggregate prefix.

The third aggregation statement matches routes to any destination, with any RD_PATH, with no distinguished attributes, and no additional attribute or local conditions.

```
AGGR {nlri_any} /.*/ dist_att_none = auto;
```

These routes are to be aggregated automatically; that is safely and algorithmically such that the aggregated NLRI does not include more NSAPs than the component routes did. By default, this statement affects the routes that are announced to all external BIS neighbors.

The fourth AGGR statement matches routes to any destination, with any RD_PATH, if they have distinguished attributes that include only CAPACITY QOS.

```
AGGR {nlri_any} /.*/ (CAPACITY security_none priority_none) = man;
```

Manual aggregation will use the longest common prefix of the specified NLRI as the aggregate route′s NLRI. This statement matches routes to any destination, so the aggregate NLRI is a ″default″ route (route with zero length NLRI). H4 id=distdis.Distribution Statement Discussion

Distribution statements are the most complex of the policy statements; they control both the selection and modification of routes for re-distribution. DIST statement processing is sequential, and like the AGGR statement, ″CONT″ and ″DONE″ affect the processing of a route by policy statements. If a DIST statement specifies ″DONE″, routes will not be affected by any subsequent DIST statements. A ″CONT″ token indicates that routes should be affected by the next DIST statement that is matched. Using the idea of increasingly or decreasingly specific <route pattern> templates in combination with ″DONE″ and ″CONT″ to selectively prohibit further processing of some routes, a wide range of policy requirements can be concisely expressed.

The first DIST statement from the example matches routes to any destination, originated by IDRP (the default match), where RD#9 is in the RD_PATH. These routes can have any distinguished attribute set (any distinguished attribute is the default match), and no additional attribute or local conditions need to be satisfied.

```
DIST {nlri_any}  /.* 9 .*/ =
 modify {allow_dist({RD#2, RD#5, RD#8});} CONT;
```

This statement does not indicate a <bis> list, so by default all external BIS neighbors′ RIB-OUTs are affected by this statement. The ″modify″ indicates that route attributes are to be modified, but the route′s ″selected status″ will remain unchanged by this DIST statement. One modification is performed which restricts the distribution of these routes (per policy F above) by altering the DIST_LISTs to only allow certain domains to receive this route.

The statement indicates ″CONT″, so these routes may be further modified, and/or selected for distribution to adjacent BIS, by subsequent DIST statements.

The second DIST statement matches routes to any destination with distinguished attributes (CAPACITY security_none priority_none).

```
DIST {nlri_any}  /.* 6/ (CAPACITY security_none priority_none)
 (CAPACITY() > 15) = {BIS#5} select_only {allow_dist(RD#5, RD#8);};
```

The {BIS#5} indication along with ″select_only″ specifies that the matched routes are to be selected for distribution only to BIS#5; an additional effect of ″select_only″ is to explicitly mark these routes as NOT distributable to all other BISs (all but BIS#5). The default action, ″DONE″, will keep these routes from being affected by other DIST statements. The ″DONE″ combined with the ″select_only″ will also prevent these routes from being matched and possibly placed in the RIB-OUT for distribution to other BISs (i.e. other than BIS#5). Using the ″allow_dist()″ function, this statement modifies the DIST_LISTs of matched routes to restrict further redistribution to domains RD#5 and RD#8.

The third DIST statement matches routes to any destination, with any RD_PATH; these routes can have any distinguished attributes, and no additional attribute or local conditions need to be satisfied.

```
DIST {nlri_any} /.*/  = select_on {init_hr();};
```

The RIB-OUTs for all neighboring BISs are affected by this statement, which selects the matched routes for distribution (″select_on″) and modifies the hierarchical recording attribute so it is initialized to ″1″ (only if the attribute is not already present and thus can be initialized according to operational procedures).

K.3.3.3 Operational Example

Consider a route with the following attributes that arrives at our BIS configured with the above Policy Information Base (PIB):

```
nlri(10:66:*) rd_path(6 22 10) hopcount(15)
```

It is a route with no distinguished attributes, which matches only one of the PREF statement route patterns:

```
PREF {nlri_any} /.*/  dist_att_none
```

and is assigned a preference of 230 (245-hopcount()) by the this PREF statement. Consider a second route to the same destination NLRI with attributes:

```
nlri(10:66:*) rd_path(1 44 9 16 10) hopcount(20)
```

It also has no distinguished attributes. It matches two PREF route patterns, however, only the first match is considered (first match).

```
PREF {nlri_any} idrp /.* 1/  dist_att_none
```

Because the route is through RD#1 it is a preferred route, and is assigned a preference of 235 (255-hopcount()). Both of these two routes are for the same set of destination NLRI; the second route, with preference value of 235 would be chosen over the route with preference value 230. If these were the only two routes to these destinations, the preferred route would be installed in our LOC_RIB and FIB.

Now aggregation policy must be considered to see how the route is to be announced. The preferred route that was placed in the LOC_RIB:

```
nlri(10:66:*) rd_path(1 44 9 16 10) hopcount(20)
```

matches one AGGR statement route pattern, which specifies automatic aggregation for those routes where it is possible:

```
1   AGGR {nlri_any} /.*/ dist_att_none = auto;
```

1 Depending on what other routes and aggregates are installed, this route may be announced individually, may
1 result in a new aggregation, or it may be part of an already instantiated aggregate. For instance, if there is
1 already an (aggregate) route to nlri(10:*), the example route could be included in the nlri(10:*) aggregate; the
1 example route would be installed in the LOC-RIB and FIB (so packets are forwarded correctly), and then a new
1 aggregate (made up of this route and the old aggregate) would be composed. If a new aggregate were to be
1 generated, a new preference value would be assigned by the PREF policy statement processing. Whether this
1 route is aggregated with other routes, or maintained individually, it must be selected by a DIST statement before
1 it will be announced.

1 Assuming that there is no aggregate for this route, it is installed in the LOC-RIB and FIB as-is, and must be
1 considered for redistribution. Again, the route:

```
1   nlri(10:66:*) rd_path(1 44 9 16 10) hopcount(20)
```

1 is matched against route patterns. It matches the first DIST statement:

```
1   DIST {nlri_any} /.* 9 .*/ =
1    modify {allow_dist({RD#2, RD#5, RD#8});} CONT;
```

1 which requires the route be modified before it is re-distributed. Applying the modifications the route becomes:

```
1  nlri(10:*) rd_path(1 44 9 16 10) dist_list_incl(2,5,8) hopcount(20)
```

1 Since this DIST statement does not terminate distribution processing, (″CONT″), other DIST statements may be
1 matched. At this point the route has been modified, but has not been selected for distribution (″modify″ rather
1 than ″select_on″ or ″select_only″ was specified). The route also matches the following DIST statement:

```
1   DIST {nlri_any} /.*/ = select_on {init_hr();};
```

1 which modifies the route (initializing the HIERARCHICAL_RECORDING attribute since it′s not already set), and
1 selects the route for distribution (to all external BIS neighbors). This DIST statement (by default) specifies
1 ″DONE″, so no further distribution processing is applied to this route. Note that other changes to the route attri-
1 butes (i.e. update of RD_PATH) will be performed as part of ″operational processing″. h4 id=defxmp.Simple
1 Default Policy

1 Among the concerns about configuring administrative policy is ease of configuration for the majority of domains
1 which may have very simple policy. A simple policy configuration must include a preference statement:

1 (Assign a preference to all routes based on the number of hops.)

```
1   PREF {nlri_any} / .*/ = 255 - hopcount();
```

1 If the routing domain will carry transit traffic, then the following minimal aggregation and distribution statements
1 are also needed:

1 (Automatic aggregation to reduce the amount of information.)

```
1   AGGR {nlri_any} / .*/ = auto;
```

1 (Select all routes for distribution; no modifications.)

```
1   DIST {nlri_any} / .*/ = select_on;
```

1 This illustrates that the configuration of the policy information base does not necessarily have to be extensive or
1 complex. A complex configuration will be the case only to the extent that the domain′s administrative policy has
1 extensive requirements and specifications.

Index

A
administrative domain 1
advertisement intervals for routes 53
aggregating routes
  See route aggregation
architectural constants 64
authentication 12, 14, 57, 58

B
BISPDU 11—22
boundary IS
  definition 3

C
CAPACITY 20, 46, 56
CEASE PDU 22, 26, 29, 30, 57, 60
checksum
  algorithm for generation 92
  encrypted, in BISPDU header 30, 94
  field in BISPDU header 12
  for RIB validation 35
  in BISPDU header 58
  unencrypted, in BISPDU header 30
confederations
  See RDC (Routeing Domain Confederation)
configuration information 23
connection management between BISs 25—30

D
Decision process
  overview of three phases 49
deployment guidelines
  for ESs and ISs 23
  for RDs 23
DIST_LIST_EXCL 17, 43, 54, 55
DIST_LIST_INCL 17, 43, 54, 55
distinguishing attributes
  derived from NPDU 61
    matching with RIB-Att 61
  distinguishing path attributes in UPDATE PDU 8
    as identifier of RIBs and FIBs 34
    type specific 38
    type-value specific 38

E
encryption 94
ENTRY_SEQ 16, 39, 40, 55
ENTRY_SET 16, 39, 40, 55, 56
error handling 57
EXPENSE 18, 45, 56, 61
EXT_INFO 16, 39, 55

F
finite state machines 25—30
flow control of BISPDUs 32
forwarding of NPDUs
  and NPDU-derived Distinguishing Attributes 60
  to external destinations 62
  to internal destinations 61

G
GDMO descriptions 65—80

H
HIERARCHICAL_RECORDING 19, 45
hold time
  of OPEN PDU 12

I
IDRP ERROR PDU 20
inter-domain link
  definition 3
  real links and inter-domain traffic 3
  virtual links and inter-domain traffic 3
  virtual links and intra-domain traffic 3
intra-domain routeing protocol vii, 3, 10, 23
ISO/IEC 10589 7

K
KEEPALIVE PDU 21, 60

L
LOCAL_PREF 16, 48
LOCALLY DEFINED QOS 18, 45, 56, 61
LSAP 17, 20

M
MULTI-EXIT_DISC 18, 44, 54, 99
multi-homed end routeing domain 3

N
NET (Network entity title)
  syntax and encoding 10
network layer reachability information
  as encoded in UPDATE PDU 20
  as related to configuration information 24
  information reduction of 54
NEXT_HOP 16, 41, 54
NLRI
  See network layer reachability information
Non-transitive attribute 15
NSAP Addresses
  syntax and encoding 10

O

OPEN PDU 12, 58
origination intervals for routes 53
overlapping routes 51

P

partial bit 15
path attributes
policies
  detecting inconsistencies 5, 48
    external inconsistencies 48
    internal inconsistencies 48
  indirect expression in UPDATE PDUs 5
  local setting of 5
  syntax and semantics of 102—114
positioning of IDRP
  with respect to ISO 8473 vii
  within the Network layer vii, 5
PRIORITY 20, 47, 56

R

RD_PATH 16, 39, 55, 56
  as encoded in UPDATE PDU 16
RD_SEQ 16, 39, 40, 55
RD_SET 16, 39, 40, 55, 56
RDC (Routeing Domain Confederation) 3, 7, 47
  associated managed object 24
  boundary of 47
  configuration information for 47
  definition 3
  disjointed. 3
  formation and deletion of 97, 101
  in OPEN PDU 13
  nested 3
  overlapping 3
  permissible policies of 47
  recursive nature of 7
  use of RDI to identify 7, 10
RDI (routeing domain identifier)
  conformance to ISO 8348/Add. 2 10
  syntax and encoding 10
real link
  See inter-domain link
reliable delivery of BISPDUs 30
RESIDUAL ERROR 18, 44, 56, 61
RIB REFRESH PDU 22, 36, 60
route aggregation
  of NLRI 55
  of path attributes 54
ROUTE-ID 16, 38, 55
ROUTE_SEPARATOR 16, 38, 48
routeing domains
  adjacent (definition) 3
  as part of global OSIE 5
  end routeing domain 3, 7
  transit routeing domain 3, 7
routes
  as expressed in UPDATE PDU 14
  definition of 8
  destinations 8
routes *(continued)*
  path attributes 8

S

SECURITY 19, 46, 56, 61
sequence numbers of BISPDUs 32
Size of BISPDUs
  maximum, of OPEN PDU 11, 12
  of the entire BISPDU 11
stub end routeing domain 3

T

tie breaking 50, 52
TR 9575 (Routeing Framework) vii, 5
TRANSIT DELAY 18, 44, 55, 61
Transitive attribute 15

U

UPDATE PDU 14, 58

V

virtual link
  See inter-domain link

W

withdrawing routes from service 14, 48