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

July 30, 1991

Additional Comments from Editor

Note: These comments from the editor are supplied to assist reviewers in locating several changes made by the editor in producing this text. It is not intended for this page to be incorporated in the final version of this international standard.

In preparing the CD-ballot text, some changes were made that were not explicitly mentioned in the associated editor's instructions, namely:

a) Since the Berlin meeting of SC6/WG2 agreed that this standard should be limited to CLNP, and further agreed that an NP should be generated to address formation of a new project for CONS-oriented inter-domain routeing, the editor has amended the title of this standard to reflect this course of action.

   Furthermore, the references to ISO 8208 and ISO 8878 have been dropped from the document references clause.

b) In line with the spirit of the associated discussions, the editor has expanded clause 7.18 so that it includes explicit text to address the requirements placed on ISO 8473 when it is operating in conjunction with IDRP.

c) In reviewing the GDMO pages, the editor noted that several timers were not listed. He has therefore added GDMO descriptions for: closeWaitDelayTimer, KeepaliveTimer, minRouteSelectionTimer, minRDOriginationTimer, and maxCPUOverloadTimer.

d) While updating the PICS for the well-known discretionary attributes, the editor noticed that there was no PICS proforma for the optional transitive path attributes of IDRP. A new PICS proforma table has been added to remedy this oversight.

e) During a session held for project editors by the ISO Central Secretariat, the editor learned that ISO/IEC presentation guidelines prohibit informative references from being included in the same clause as normative ones. In particular, documents that are only mentioned in passing, and which do not impose normative requirements on IDRP should not be listed in the ″References″ clause. Therefore, the editor has rearranged this clause into 2 parts (Normative References and Informative References), and has classified the references accordingly.

Contents

Figures

Tables

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[1]) 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.[2]

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] See clause 5 for a listing of symbols and abbreviations.

[2] 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 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[3] 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 databases 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[4]**,** *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*

[3] An Intermediate system that implements this protocol is known as a Boundary Intermediate system (BIS).

[4] To be published

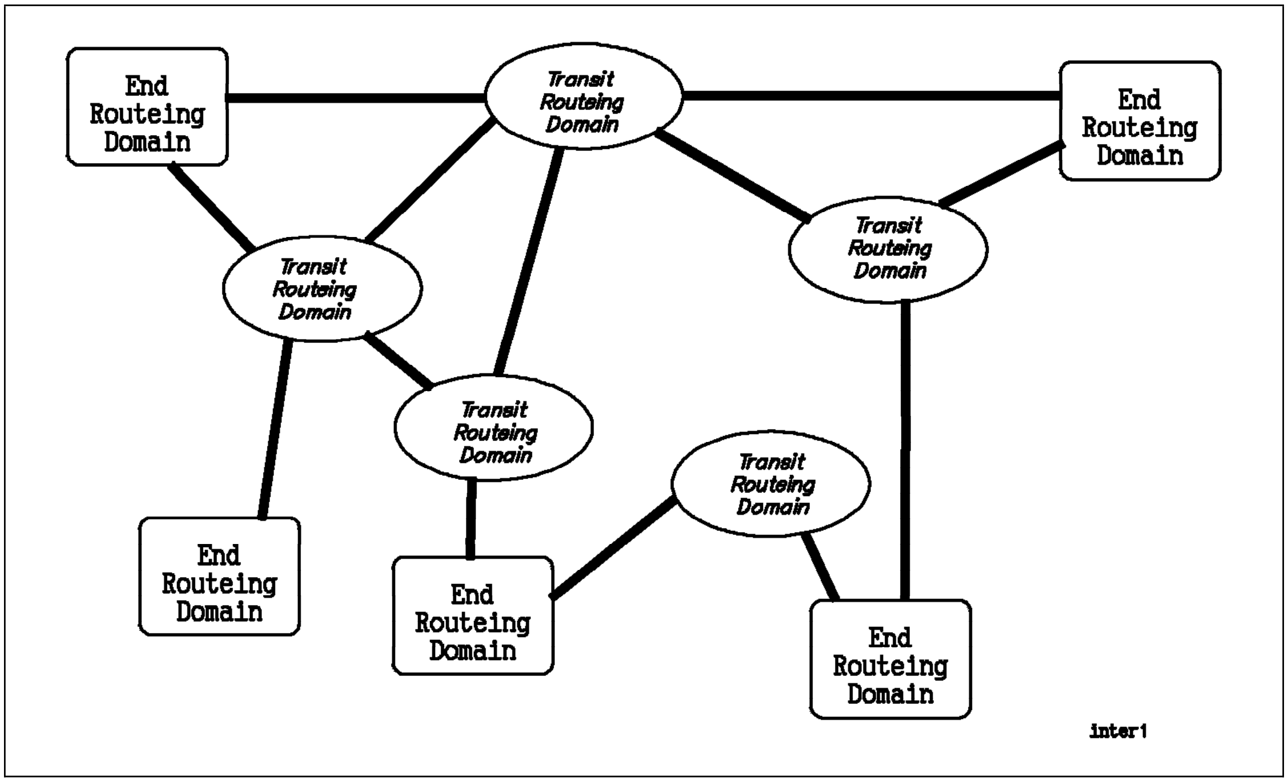

Figure 1. Field of Application. The Inter-domain Routeing Protocol operates between routeing domains; intra-domain routeing is not within its scope.

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 DTR 9577[4], *Information Processing Systems - Telecommunications and Information Exchange between Systems - Protocol identification in the Network Layer*

ISO TR 9575: 1989, *Information Processing Systems - Telecommunications and Information Exchange between Systems - OSI Routeing Framework*

ISO PDTR 10172[4], *Information Processing Systems - Data Communications - Network/Transport Interworking Specification*

ISO 10589[4], *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)*

3. Informative References

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)*

4. Definitions

4.1 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

4.2 Network Layer Architecture Definitions

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

a) End system
b) Intermediate System
c) Subnetwork

4.3 Network Layer Addressing Definitions

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

a) Subnetwork point of attachment

4.4 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

4.5 Intra-domain Routeing Definitions

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

a) Adjacency
b) Link

4.6 Network/Transport Protocol Interworking Definitions

This International Standard uses the following terms defined in ISO DTR 10172:

a) Interworking Function Unit

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

Inter-domain link: A link between two or more Boundary Intermediate systems (see Figure 2):

— A link between two BISs in the same routeing domain may be either a real (physical) link or a virtual (logical) link. Links between BISs in the same routeing domain may optionally be used to carry intra-domain traffic as well as inter-domain traffic.
— A link between two BISs located in adjacent routeing domains must be a real (physical) link, and may not be used to carry 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 8.14 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

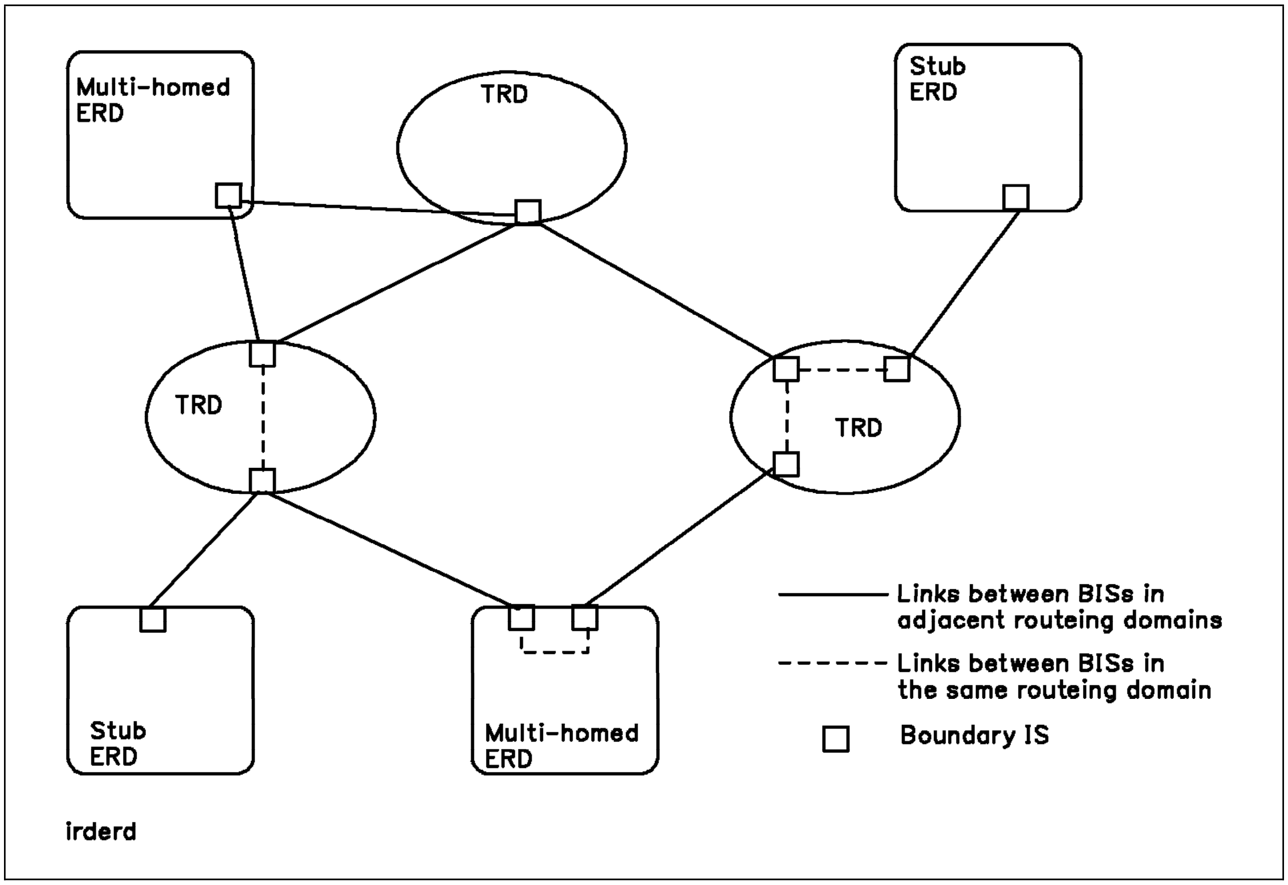
Multi-homed ERD
TRD
Stub ERD
TRD
TRD
Stub ERD
Multi-homed ERD
Links between BISs in adjacent routeing domains
Links between BISs in the same routeing domain
Boundary IS
irderd

Figure 2. Intermediate Routeing Domains and End Routeing Domains. The classification of a routeing domain as an TRD or an ERD depends upon both its interconnections to other routeing domains and its relaying policies with respect to them.

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.

5. Symbols and Abbreviations

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

5.1 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

5.2 Addressing Abbreviations

AFI	Authority and Format Identifier
DSP	Domain Specific Part

IDI	Initial Domain Identifier
IDP	Initial Domain Part
NET	Network Entity Title
NPAI	Network Protocol Address Information
NSAP	Network Service Access Point
SNPA	Subnetwork Point of Attachment

5.3 Other Abbreviations

BIS	Boundary Intermediate System
CL	Connectionless Mode
CLNS	Connectionless Mode Network Service
CM	Confederation Member
CO	Connection-oriented Mode
CONS	Connection-oriented Network Service
ERD	End Routeing Domain
ES	End System
FIB	Forwarding Information Base
IDRP	Inter-domain Routeing Protocol (an acronym for the protocol described in this international standard)
MIB	Management Information Base
NLRI	Network layer reachability information
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
TRD	Transit Routeing Domain

6. 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 7), 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 8.5; methods for establishing BIS-BIS connections are described in clause 8.6.

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.

b) **BISPDU-Send Process:** responsible for constructing BISPDUs which contain control and routeing information which is 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 databases.

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 8.10).

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.

6.1 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.

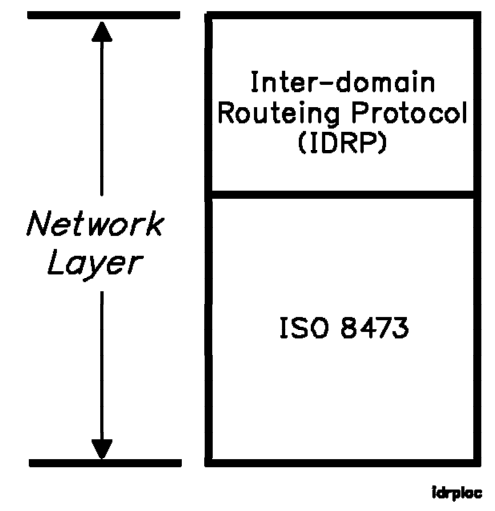

Figure 3. Position of IDRP within Network Layer

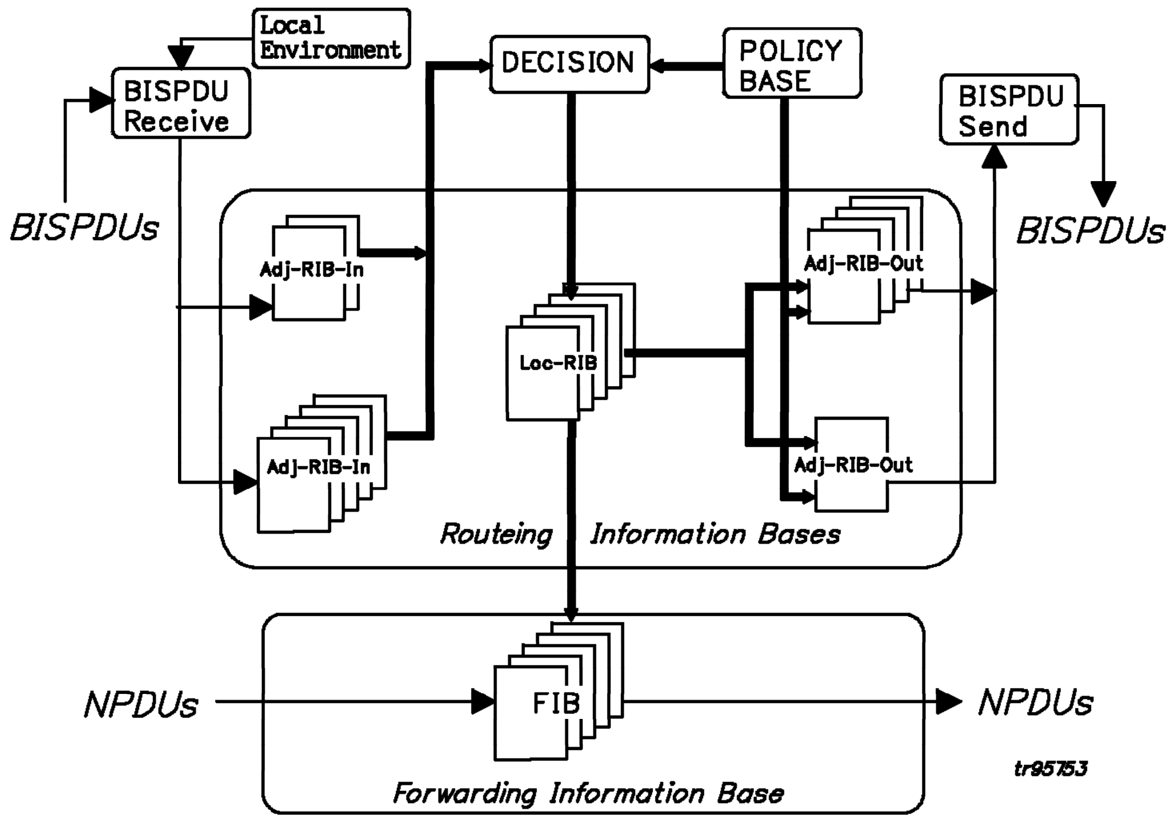

Figure 4. Inter-domain Routeing Components

6.2 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. However, the semantics of routeing policies and the methods for establishing them are outside the scope of this standard.[5]

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[6] for potential paths. 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.

— provides methods to detect mutually unsatisfiable multi-lateral policies (between a local RD and several other RDs).

6.3 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.[7]

6.4 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.

6.5 Routeing Domain Confederations

IDRP provides support for Routeing Domain Confederations (RDCs); this allows 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 sets of routeing domains (which may themselves form an RDC). Thus, the definition of an RDC is recursive: a confederation member may be a single routeing domain or another confederation.

The existence of an RDC is visible only to members of the confederation. To RDs outside the confederation, an RDC will be indistinguishable from an individual routeing domain.

6.6 Routes: Advertisement and Storage

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

— *Routes* are advertised in UPDATE PDUS: the *destination* is the systems whose NSAPs 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. Those routes that will be advertised to other BISs must be present in the local BIS's Loc-RIBs and its Adj-RIBs-Out, and the next hop for each of these routes must also be present in the local BIS's Forwarding Information Bases.

A BIS can support multiple routes to the same destination by maintaining multiple RIBs and the corre-

[5] Appendix G provides an illustration of a policy description method and its associated semantics as one example of how policies might be expressed.

[6] *Degree of preference* is discussed in clause 8.17.3.

[7] 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 10589.

sponding multiple FIBs. Each Loc-RIB will be identified by a different RIB-Att (see clauses 6.7 and 6.8), and can contain only 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.

Multiple routes that have identical path attributes but different destinations may be combined into a single UPDATE PDU. Routes with different path attributes—even if to the same destination—can either be carried in different PDUs, or the routeing information may be aggregated according to clause 8.17.5.

6.7 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 permitted combinations of distinguishing attributes; this in turn limits the number of RIBs and FIBs that a BIS can support. Although this number is large, it will be limited in practice by the number of useful combinations of these attributes. 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 8.10.1).

6.8 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 it receives an UPDATE PDU, a BIS must unambiguously determine the Adj-RIB-In which will be used to store the routeing information. The local system examines the path attributes in an UPDATE PDU: the set of distinguishing path attributes that are present in the UPDATE PDU map into a RIB-Att. In turn, the RIB-Att identifies the routeing information base, as in clause 8.1.2.2

When it receives an NPDU, a BIS 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 9.2 and 9.3).

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

6.9.1 Internal Neighbor BIS

Each BIS establishes and maintains communications with all other BISs located in its routeing domain. BISPDUs flow over inter-domain links, which may be either real or virtual. The identity of all BISs within a routeing domain is contained in managed object **INTERNAL_BIS** described in clause 8.3.

6.9.2 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 4.7: 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 a single common subnetwork. The identity of the neighbor BISs located in adjacent routeing domains is contained in managed object **EXTERNAL-BIS-NEIGHBORS** described in clause 8.3.

6.10 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.

6.11 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 be structured as described in clause 8.1.

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.

6.12 Functional Organization of the Protocol

The protocol specified in this International Standard may be organized into two sublayers: network independent functions and network dependent functions.

The network independent functions provide reliable transfer of routeing information between adjacent BISs over a full-duplex BIS-BIS connection. These functions are independent of the specific network service operating immediately below this sublayer. The network independent functions are described in clause 8.

The network dependent functions provide the interface between IDRP and ISO 8473; service primitives are used to communicate between IDRP and ISO 8473, as described in clause 10.

6.13 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.

6.13.1 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, DIS 10589

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 databases.

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.

6.13.2 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 Domains 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.

Interworking Function: It neither specifies the details of interworking functions between CONS and CLNS, nor does it specify the BISs in which such functions must be implemented. These matters are left as a local decision for the administrators of routeing domains for which interworking may be needed.

Suppression of Transient Loops: Although it provides mechanisms to detect and suppress looping of BISPDUs, it provides no mechanisms to detect or suppress transient looping of either ISO 8473 NPDUs or ISO 8208 CR packets.

7. 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).

The types of PDUs used by this protocol are:

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

7.1 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)
Version (1 octet)
BISPDU Type (1 octet)
Sequence (4 octets)
Acknowledgement (4 octets)
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 31 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.

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

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

BISPDU Type	Code
CEASE PDU	6
RIB REFRESH PDU	7

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 8.5.2.

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.

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 be used to a checksum on the content of the BISPDU (see 8.8 and Appendix B), or a checksum with authentication of the source of the BISPDU (see 8.9). The authentication function is achieved by encryption of the basic checksum.

7.2 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 sequence field of the OPEN PDU contains the sender′s initial sequence number (ISN). The Acknowledgement field of the OPEN PDU is set to zero, since there are no previous PDUs to be acknowledged.

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

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

The meaning and use of these fields are as follows:

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, RIB CHECKSUM PDUs, 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 is able to handle in an incoming BISPDU. Every BIS is required to support a minimum of **MinBISPDULength** octets.

Outstanding PDUs: This field contains a one octet unsigned integer that is the maximum number of BISPDUs that may be sent to this BIS without receiving an acknowledgment.

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 BIS which is sending this PDU.

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 remote BIS (that is, the one that receives this OPEN PDU). That is, it contains an encoding of all or part of the information contained in managed object **RIB-AttsSet** . (See clauses 8.3 and 8.10.1.)

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 8.19.2.

The encoding of this field is as follows:

Number of Non-empty RIB-Atts
Number of Distinguishing Attributes in First Set
First attribute, first set
....
Last Attribute, first set
Number of Distinguishing Attributes in Second Set
First attribute, second set
....
Last Attribute, second set
...

Number of Distinguishing Attributes in Nth Set
First attribute, Nth set
....
Last Attribute, Nth set
...
Number of Distinguishing Attributes in Last Set
First attribute, last set
....
Last Attribute, last set

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 8.10.1.), 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.

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

— a type specific distinguishing attribute (see clause 8.12.2) is encoded as a single octet, using the type code specified in clause 7.3 for the corresponding UPDATE PDU path attribute.

— a type-value-specific distinguishing attribute (see clause 8.12.2) is encoded as a triple <type, length, value>, using the type code, length, and the first four fields of the value field, as specified in clause 7.3 for the corresponding UPDATE PDU path attribute.

Note: The first four fields of the source- and destination-specific attributes list the NSAP address to which the attribute applies and the value of the attribute. Although additional fields may be present (for example, to list metrics), they are not carried in the OPEN PDU.

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 as a single prefix, according to clause 8.1.2.1.

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 covering the contents of that BISPDU. Its use is described in clause 8.5.1.

b) Code 2 indicates that the Validation Pattern field in the header of each BISPDU contains an encrypted checksum covering the contents of that BISPDU. Encryption of the checksum provides the authentication function for the source of the BISPDU. Its use is described in clause 8.9.

Authentication Data: The form and meaning of this field is a variable-length field that depends on the Authentication Code, as described in clause 8.9. The length of the authentication data field can be determined from the Length field of the BISPDU header.

7.3 UPDATE PDU

UPDATE PDUs are used to exchange routes (see 6.6) between pairs of BISs. The UPDATE PDU contains a fixed header and several additional fields, structured as shown in Figure 5. A single UPDATE PDU may contain several path attributes, each of which is encoded as a 4-tuple as described below. All path attributes contained in a given UPDATE PDU apply to the Network Layer Reachability Information which is contained in the same PDU. Note that all fields may have arbitrary alignment:

Fixed Header
Total Path Attributes Length (2 octets)
Path Attributes (variable)
Network Layer Reachability Information (variable)

The use of these fields is as follows:

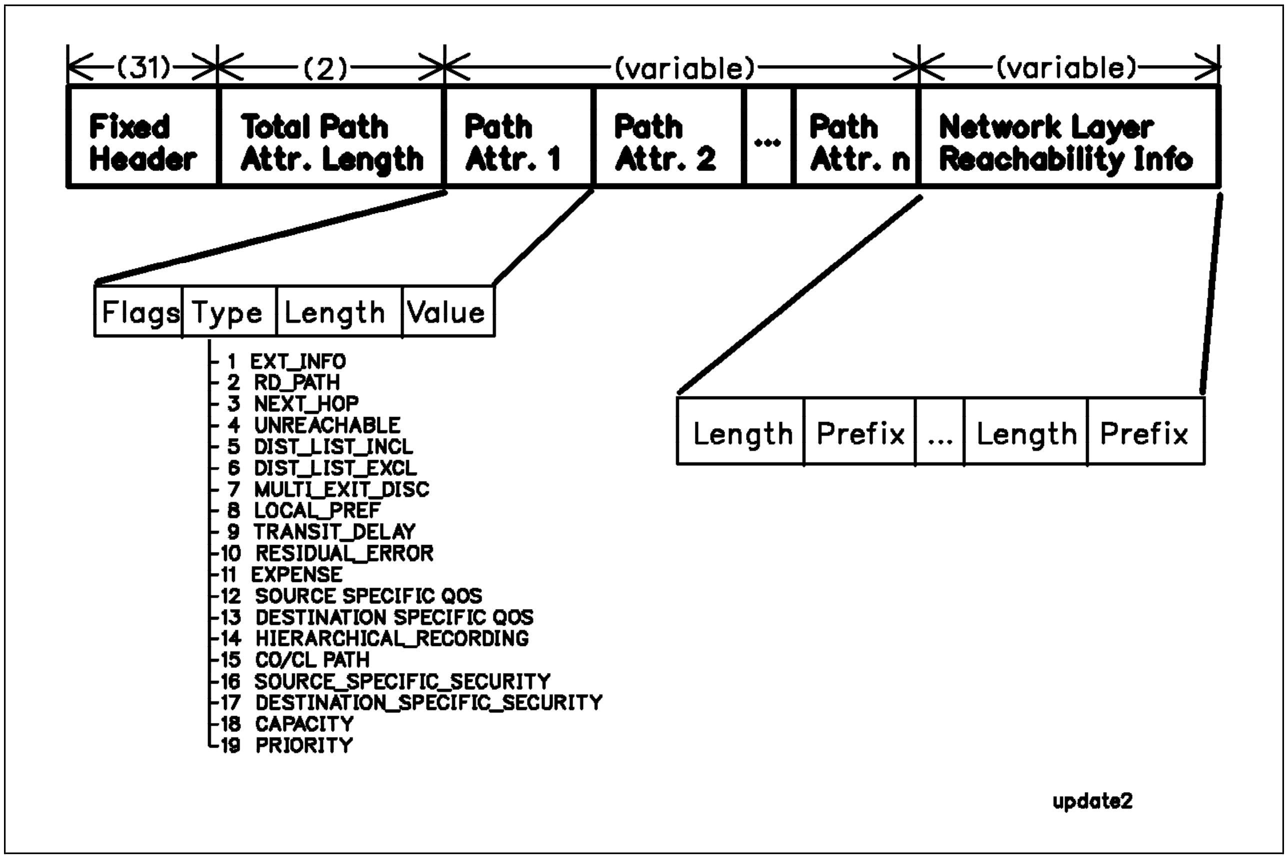

Figure 5. Structure of the UPDATE PDU

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. An UPDATE PDU may contain more than one path attribute. 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 information contained in 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 8.12.

— **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 attributes field contain the values of the attributes, which are interpreted according to the attribute flags and the attribute type code. The supported attribute values and their uses are the following:

a) EXT_INFO (Type Code 1):

The EXT_INFO attribute 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 8.13.1

b) RD_PATH (Type Code 2):

The RD_PATH attribute is composed of a series of RD path segments. Each RD 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:
  - **1-** RD_SET - unordered set of RDIs that the UPDATE PDU has traversed.
  - **2-** RD_SEQUENCE - ordered set of RDIs that the UPDATE has traversed.
- 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 doublets <length, RDI>: the length of field contains the length RDI in octets, and the RDI itself is encoded according to clause 8.1.

Usage of this attribute is defined in clause 8.13.2.

c) NEXT_HOP (Type Code 3):

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

1) to permit a BIS to advertise a different BIS′s NET in the ″NET of Next Hop″ field
2) 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)
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.

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 8.13.3.

d) UNREACHABLE (Type Code 4):

The UNREACHABLE attribute has a length of zero octets; its presence is used to notify a BIS peer that path described by the path attributes of this UPDATE PDU can not be used to reach the destinations listed in the NLRI field of this UPDATE

PDU. There are several reasons why a route will be declared to be unreachable:

1) An originating routeing domain administrator no longer controls some or all of the systems that were previously advertised in the NLRI field of an UPDATE PDU.
2) Local policy has decided that a previously available route is no longer available for use by those to whom it had been advertised.
3) The BIS-BIS connection with a neighbor BIS has been terminated.
4) It was learned of by means of an UPDATE PDU that contained the UNREACHABLE ATTRIBUTE.

Its usage is described in clause 8.13.4.

e) DIST_LIST_INCL (Type Code 5):

The DIST_LIST_INCL attribute 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. The list is expressed in terms of RDI prefixes. Its usage is described in clause 8.13.5.

The list of RDIs is encoded as one or more triples of the form <type, length, value>, whose fields are described below.

Type (1 octet)
Length (1 octet)
Value (variable)

The use and meaning of these fields are as follows:

1) Type:

   A type of 1 indicates that the value field contains an RDI prefix.

2) Length:

   The length field indicates the length in semi-octets of that prefix.

3) Value:

   The value field contains the RDI prefix, encoded according to clause 8.1.2.1.

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

f) DIST_LIST_EXCL (Type Code 6):

The DIST_LIST_EXCL attribute 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. The list is expressed in terms of RDI prefixes. Its usage is described in clause 8.13.6.

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

The list of RDIs is encoded as one or more triples of the form <type, length, value>, whose fields are described below.

Type (1 octet)
Length (1 octet)
Value (variable)

The use and meaning of these fields are as follows:

1) Type:

   A type of 1 indicates that the value field contains an RDI prefix.

2) Length:

   The length field indicates the length in semi-octets of that prefix.

3) Value:

   The value field contains the RDI prefix, encoded according to clause 8.1.2.1.

g) MULTI-EXIT_DISC (Type Code 7):

The MULTI-EXIT_DISC attribute (Multi-exit Discriminator) 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 8.13.7.

h) LOCAL_PREF (Type Code 8):

The LOCAL_PREF attribute is 1 octet in length. It is used by a BIS to inform other BISs in its own RD of the originating BIS's degree of preference for an advertised path. Usage of this attribute is described in clauses 8.13.8 and 8.15.2.

i) TRANSIT DELAY (Type Code 9)

The TRANSIT DELAY attribute 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 8.13.9.

j) RESIDUAL ERROR (Type Code 10)

The RESIDUAL ERROR attribute 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 8.13.10.

k) EXPENSE (Type Code 11)

The EXPENSE attribute 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 8.13.11.

l) SOURCE SPECIFIC QOS (Type Code 12)

This variable length field contains the parameters associated with ISO 8473′s source address specific QOS parameters, and 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:

1) NSAP prefix length:

   Gives the length in semi-octets of the NSAP prefix

2) NSAP prefix:

   Contains the NSAP prefix, encoded according to clause 8.1.2.1. If an ISO 8473 NPDU′s source NSAP address matches the NSAP prefix, then it will be routed according to the indicated QOS value.

3) QOS length:

   Gives the length in octets of the QOS value field.

4) QOS value:

   Contains the value of the QOS parameter.

5) Metric length:

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

6) Metric value:

   Contains the value of the metric associated with the path being advertised (that is, its ″cost″)

Its usage is defined in clause 8.13.12.

m) DESTINATION SPECIFIC QOS (Type Code 13)

This variable length field contains the parameters associated with ISO 8473′s destination address specific QOS parameters, and 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:

1) NSAP prefix length:

   Gives the length in semi-octets of the NSAP prefix

2) NSAP prefix:

   Contains the NSAP prefix, encoded according to clause 8.1.2.1. If an ISO 8473 NPDU′s destination NSAP matches the NSAP prefix, then it will be routed according to the indicated QOS value.

3) QOS length:

   Gives the length in octets of the QOS value field.

4) QOS value:

   Contains the value of the QOS parameter.

5) Metric length:

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

6) Metric value:

   Contains the value of the metric associated with the path being advertised (that is, its ″cost″)

Its usage is defined in clause 8.13.13.

n) HIERARCHICAL RECORDING (Type Code 14)

This is 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 8.13.14.

— CO/CL PATH (Type Code 15)

The CO/CL attribute has a length of 1 octet, and is used to indicate whether the RD_PATH is connectionless mode or connection mode, and whether it includes any IFUs. When bit 1 of this field is equal to 1, the mode of the RD_PATH is connectionless; when it is equal to 0, the mode is connection-oriented. When bit 8 of this attribute is equal to 1, the UPDATE PDU has passed through at least one IFU; when it is equal to 0, the UPDATE PDU has passed through no IFUs. Bits 2 through 7 of this field are reserved.

Note: The methods by which the mode of the RD_PATH is determined are outside the scope of IDRP.

Its usage is defined in clause 8.13.15

— RD_HOP_COUNT (Type Code 16)

The RD_HOP_COUNT is 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 8.13.16.

— SOURCE SPECIFIC SECURITY (Type Code 17)

This variable length field contains the parameters associated with ISO 8473's source address specific security parameter, and is encoded as follows:

NSAP prefix length (1 octet)
NSAP prefix (variable)
Length (1 octet)
Security level (variable)

The use and meaning of the fields is as follows:

a) NSAP prefix length:

Gives the length in semi-octets of the NSAP prefix

b) NSAP prefix:

Contains the NSAP prefix, encoded according to clause 8.1.2.1. If an ISO 8473 NPDU's source NSAP address matches the NSAP prefix, then it will be routed according to the indicated security level.

c) Length:

Gives the length in octets of the Security level field.

d) Security level:

Contains the value of the security level.

Its usage is defined in clause 8.13.17.

— DESTINATION SPECIFIC SECURITY (Type Code 18)

This variable length field contains the parameters associated with ISO 8473's destination address specific security parameter, and is encoded as follows:

NSAP prefix length (1 octet)
NSAP prefix (variable)
Length (1 octet)
Security level (variable)

The use and meaning of the fields is as follows:

a) NSAP prefix length:

Gives the length in semi-octets of the NSAP prefix

b) NSAP prefix:

Contains the NSAP prefix, encoded according to clause 8.1.2.1. If an ISO 8473 NPDU's destination NSAP matches the NSAP prefix, then it will be routed according to the indicated security level.

c) Length:

Gives the length in octets of the security level field.

d) Security level:

Contains the value of the security level.

Its usage is defined in clause 8.13.18.

— CAPACITY (Type code 18)

The CAPACITY attribute 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 8.13.19.

— PRIORITY (Type Code 19)

The PRIORITY attribute 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 0000 0001 through 0000 1110 indicate higher priorities. A particular value of this parameter, *m*, means that the BIS will not forward any ISO NPDUs whose priority parameter has a value less than *m*. Detailed usage rules are presented in clause 8.13.20.

Network Layer Reachability Information: This variable length field contains a list of NSAP address prefixes. The length in octets of the *Network Layer Reachability Information* is not encoded explicitly, but can be calculated (see Figure 5) as:

BISPDU Length – 33 – *Total Path Attributes Length*,

where *BISPDU Length* is the value encoded in the Fixed Header, *Total Path Attributes Length* is the value encoded in the variable part of the UPDATE PDU, and 33 octets is the combined length of the *Fixed Header* field and the *Total Path Attributes Length* field.

Reachability information is encoded as one or more 2-tuples of the form <length, prefix>, whose fields are described below.

Length (1 octet)
Prefix (variable)

The use and meaning of these fields are as follows:

a) Length:

The length field indicates the length in semi-octets of following prefix field A length of zero indicates a prefix that matches all NSAPs.

b) Prefix:

The value field contains the NSAP prefix, encoded according to clause 8.1.2.1.

7.4 IDRP ERROR PDU

IDRP ERROR PDUs are sent when an error condition is detected. The BIS connection will be closed in the following manner upon the detection of an error condition:

a) the BIS that detects the error (the local BIS) shall issue a IDRP ERROR PDU,

b) upon receipt of the IDRP ERROR PDU, the remote BIS shall return a CEASE PDU to the local BIS

c) upon receipt of the CEASE PDU from the remote BIS, the local BIS shall close the connection.

In addition to its fixed header, the IDRP ERROR PDU contains the following fields:

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

The use of these fields is as follows:

Error code: The Error code field is 1 octet long. 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

All errors are fatal to the BIS connection.

Error subcode: The Error subcode is one octet long. It must be present if an OPEN_PDU_Error or an UPDATE_PDU_Error is being reported. It is not present for other types of errors.

The error subcode, if present, 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:

— OPEN PDU Error subcodes:

Subcode	Value
Unsupported_Version_Number	1
Bad_Max_PDU_Size	2
Bad_Outstanding_PDUs	3
Bad_Peer_RD	4
Unsupported_Authentication_code	5
Authentication_Failure	6
Bad_RIB-AttsSet	7
RDC_Mismatch	8

— 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

Data: This variable length field contains zero or more octets of data to be used in diagnosing the reason for the NOTIFICATION. 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).

7.5 KEEPALIVE PDU

BISs use the periodic exchange of KEEPALIVE PDUs to determine if peers are 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.

KEEPALIVE PDU consists of only a PDU header and has a length of 31 octets.

7.6 CEASE PDU

To ensure the delivery of a IDRP ERROR PDU, this protocol requires the BIS that receives a IDRP ERROR PDU to send back a CEASE PDU and then wait for one retransmission time for that CEASE PDU to be acknowledged. In addition, in absence of any fatal errors, a BIS peer may choose at any given time to close its peer connection by sending the CEASE PDU.

The CEASE PDU consists of only a PDU header and has length of 31 octets.

7.7 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
Op COde (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 is empty except for the OpCode RIB REFRESH Request. In this case, it 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 8.10.3.

8. Network Independent Protocol Functions

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.

8.1 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/AD2.

8.1.1 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 8.2.2, the protocol specified in this International Standard will operate correctly.

8.1.2 NSAP Address Prefixes

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

Note: 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.

8.1.2.1 Encoding of Prefixes

The encoded form of an NSAP address prefix that is carried in a BISPDU shall be 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.

An NSAP prefix is said to be of length ″n″ when it is is constructed from the first ″n″ characters of the underlying NSAP address, expressed in its abstract syntax.

8.1.2.2 Prefix Matching

As part of the forwarding process, a BIS must match a destination NSAP address against NSAP address prefixes. 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 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 (or NSAP′) contains fewer semi-octets than the NSAP address prefix, then there is no match.

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

Note: 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.

8.2 Deployment Guidelines

8.2.1 Deployment of RDs

To participate in this inter-domain routeing protocol:

— Each routeing domain must contain at least one Boundary Intermediate system

— Each Transit routeing domain must support inter-domain links to at least two different routeing domains (to allow it to provide a relaying function between external routeing domains)

— Each routeing domain that contains End systems must have at least one BIS capable of delivering NPDUs to the intra-domain routeing function (so that they can be routed to internal destinations via an intra-domain routeing function)

8.2.2 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 an UPDATE PDU originated by a given RD should be associated with only one 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 generate UPDATE PDUs 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 8.1.2.2, 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.

8.3 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 reachable via a single subnetwork hop. This information is contained in managed object **EXTERNAL-BIS-NEIGHBORS**, which consists if 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 is a list 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 8.10.1, 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.

8.4 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 8.2.2, thus assuring correct operation of this protocol.

c) For efficient operation of this protocol, the UNREACHABLE attribute should not be used to report the NLRI of systems in the local routeing domain that are offline. The UNREACHABLE attribute 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.

d) 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.

Note: 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 reject UPDATE PDUs that contain an excessive amount of NLRI information or that can cause inefficient operation of this protocol.

8.5 Reliable Delivery 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 8.6. It uses the mechanisms described below to provide reliable delivery of BISPDUs between pairs of BISs communicating over a connection.

8.5.1 Checksum over BISPDU

A checksum that covers the contents of each BISPDU is always present in the Validation Pattern field of that BISPDU. If the Authentication Type code of the OPEN PDU was 2, the Validation Pattern field contains an encrypted checksum generated in accordance with clause 8.9.; if it was 1, then the Validation Pattern field contains an unencrypted checksum generated as defined below. A BISPDU received with an incorrect checksum (either encrypted or unencrypted) shall be discarded without any further action.

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 Appendix B 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 Appendix B 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.

8.5.2 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. Sequence numbers do not wrap.

The rules for manipulating sequence numbers are:

— When the FSM for a given BIS-BIS connection is initialized, it shall start with its sequence number equal to 1.

— The sequence number shall be incremented by 1 each time that a data-carrying BISPDU is transmitted. The data-carrying BISPDUs are: OPEN PDU, UPDATE PDU, and RIB REFRESH PDU.

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

— If a BIS-BIS connection is broken and subsequently re-established without re-initializing the associated FSM, the next sequential sequence number shall be used.

— After it sends a BISPDU with the maximum permissible sequence number ($2^{32} - 1$), a BIS shall tear down the BIS-BIS connection and then establish a new connection which shall start with sequence number equal to 1.

   Note: 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.

8.5.3 Retransmission

To detect BISPDUs that have been lost or discarded, the sending BIS shall set a retransmission timer for each data-carrying BISPDU sent. The timer shall be set to a value that will allow a return acknowledgement to be received in a reasonable amount of time. If an acknowledgement is received before the timer expires, the timer is cleared. If the timer expires before receipt of an acknowledgement, the BISPDU is retransmitted and the timer is restarted.

Note: Retransmission timer values should be dynamically calculated based on the round trip time characteristics of the BISPDU connection.

8.5.4 Flow Control

This protocol uses a flow control scheme based on the sender′s current number of unacknowledged BISPDUs and the maximum number of unacknowledged BISPDUs permitted by the receiver.

A windowing scheme is used by the receiver to determine acceptable sequence numbers:

— The left edge of the window is the number of the last acknowledged sequence number plus one

— The right edge of the window is the sum of the left edge plus the allowed maximum number of outstanding BISPDUs, as included in the Outstanding BISPDUs field of the OPEN PDU.

   The windowing scheme does not apply to the CEASE PDU or the IDRP ERROR PDU. That is, these BISPDUs shall be received and processed regardless of their sequence numbers.

The right edge of the window is the first sequence number that can not be sent or received, and the total window size is equal to the value in the Outstanding PDUs field of the OPEN PDU.

A transmitting system may send a data-carrying BISPDU (see 8.5.2) without waiting for an acknowledgement until it reaches the Outstanding BISPDUs limit of the receiving system. At that point, new BISPDUs shall not be sent until an acknowledgement is received that advances the left edge of the window.

When a BIS receives a BISPDU with a sequence number equal to the left edge of its window, it shall acknowledge that BISPDU and shall advance both edges of its window by one.

When a BIS receives a BISPDU with a sequence number within its window but not equal to the window's left edge, it shall save that BISPDU if buffering is available. If a subsequent BISPDU is received that advances the window, a check shall be made to see if any saved BISPDUs in the queue can advance the receive window. When there are several BISPDUs that can advance the window, they shall be presented to the Update-Receive Process in the following order: IDRP ERROR PDUs (highest sequence number first), CEASE PDUs (highest sequence number first), other BISPDUs (in order of increasing sequence numbers).

Note: Acknowledgements should be made in a timely fashion for as many BISPDUs as possible in order to allow proper functioning of algorithms to compute dynamic retransmission timers.

8.5.5 Acknowledgements

To guarantee the ordered delivery of BISPDUs, this protocol uses a positive acknowledgement and retransmission mechanisms. Acknowledgement is cumulative: by acknowledging a particular BISPDU, the system is affirming that all BISPDUs with lower sequence numbers have been correctly received and accepted. The acknowledgement process is as follows:

— KEEPALIVE PDUs, CEASE PDUs, and damaged PDUs shall not be acknowledged.

— An OPEN PDU, UPDATE PDU, or RIB REFRESH PDU shall be acknowledged when it is received without error and all of the following conditions are true:

  a) its sequence number lies within the window defined in clause 8.5.4

  b) all BISPDUs with sequence numbers falling between the left edge of the window and the sequence number of this BISPDU have been received without error.

Acknowledgements can be carried in the headers of any type of BISPDU. A KEEPALIVE PDU may be used if there are no data-carrying BISPDUs to carry the return acknowledgement. Acknowledgement should be made in a timely fashion to aid the sending BIS in calculating dynamic retransmission timers. BISPDUs are retransmitted when there is no timely acknowledgement by the receiving system.

If a BISPDU is received with errors, then it should be handled according to the methods described in clause 8.19.

8.6 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. An example of managing such a communications channel is shown in Appendix N in the form of a listing of pseudo-code.

8.6.1 BIS Finite State Machines

A BIS-BIS connection will progress through a series of states during its lifetime. The states are shown in Figure 6, and are individually described below. In this figure, the boxes represent the states of a BIS Finite State Machine (FSM) and the arcs represent changes in state. Each arc is annotated to show the event causing the state change (shown above the line) and the resulting output (shown below the line).

A BIS shall maintain one FSM for each BIS-BIS connection that it supports, and each FSM in a given BIS shall run independently of one another.

8.6.2 CLOSED State

The CLOSED state exists when no BIS-BIS connection exists and there is no connection record allocated.

8.6.3 Active States

During its lifetime, a BIS-BIS connection will progress through a series of states which are described below.

Initially the BIS Finite State Machine is in the CLOSED state. Connections are opened by generating a Start event which triggers an OPEN PDU that attempts to establish a connection with a specific BIS, which must be one of those whose NET is reported in either managed object **INTERNAL-BIS** or managed object **EXTERNAL-BIS-NEIGHBORS**. Similarly, inbound connection requests will be accepted only from BISs whose NETs are reported in one of these two managed objects.

8.6.3.1 OPEN-SENT State

The OPEN-SENT state is entered after processing a Start event. The local BIS shall allocate a connection record, shall generate an initial sequence number, and shall send an OPEN PDU to the remote BIS. The local BIS shall remain in the OPEN-SENT state until one of the following events occurs:

a) If an OPEN PDU is received from the remote BIS without an acknowledgement of the OPEN PDU sent by the local system, then there have been simultaneous attempts to open a connection, with the OPEN PDUs passing each other in transit:

  — If the incoming OPEN PDU successfully passed local error checking, then the local BIS shall resend its own OPEN PDU with the same sequence number, but shall also

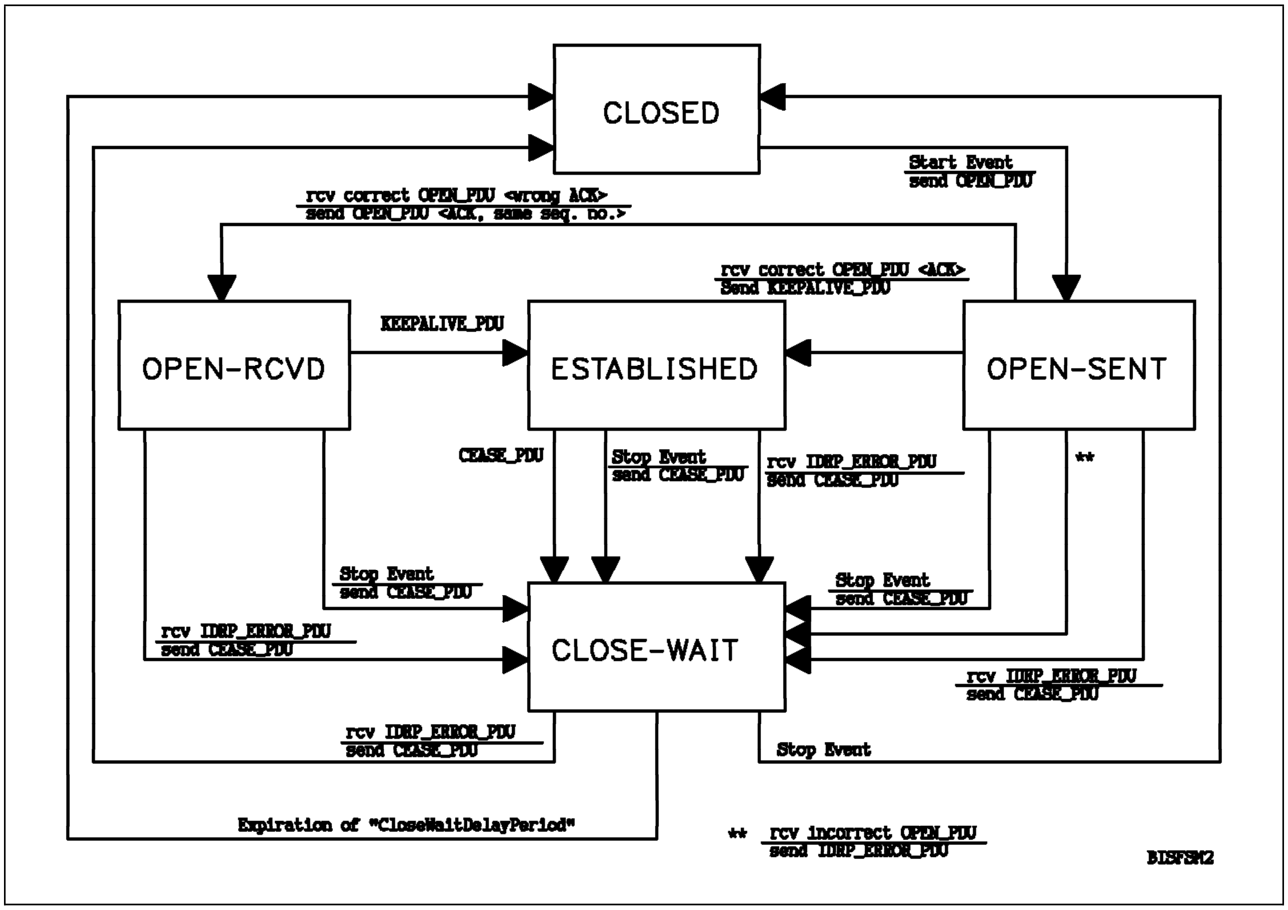

Figure 6. BIS Finite State Machine Diagram

include an acknowledgement of the remote BIS's OPEN PDU. It shall then change its local state to OPEN-RCVD.

— If the incoming OPEN PDU failed local error checking, and the local error checking requires the local BIS to send the IDRP ERROR PDU, then the local system shall send the IDRP ERROR PDU with the same sequence number used for its previously issued OPEN PDU to that BIS, and shall also include an acknowledgement of the remote BIS's OPEN PDU. Regardless of whether IDRP ERROR PDU is sent or not, the local system then changes its state to CLOSE-WAIT.

b) If the OPEN PDU received from the remote BIS acknowledges the local BIS's OPEN PDU, then:

— if the incoming OPEN PDU successfully passed local error checking, the local system shall acknowledge the incoming OPEN PDU by sending the KEEPALIVE PDU. The local BIS shall change its state to ESTABLISHED.

— if the incoming OPEN PDU fails to pass the local error checking, the local system shall send the IDRP ERROR PDU (if required by the local error checking procedure) to acknowledge the receipt of the OPEN PDU. The local BIS shall then change its state to CLOSE-WAIT.

c) If the local system receives an IDRP ERROR PDU that acknowledges receipt of its OPEN PDU, then the local system shall send a CEASE PDU, and shall change its state to CLOSE-WAIT.

d) If the local system generates a Stop event, it shall send a CEASE PDU to its peer, and shall then enter the CLOSE-WAIT state.

8.6.3.2 OPEN-RCVD State

The OPEN-RCVD state is entered from the OPEN-SENT state after the local BIS has sent out its OPEN PDU. The local BIS state shall wait in OPEN-RCVD state for the remote BIS to send either a KEEPALIVE PDU or an IDRP ERROR PDU to acknowledge receipt of the local BIS's OPEN PDU. Upon receipt of an acknowledgement, the local BIS shall take the following actions:

— If the local system receives an IDRP ERROR PDU that acknowledges the OPEN PDU sent by the local system, then the local system shall send a CEASE_ PDU, and shall change its state to CLOSE-WAIT.

— If the local system receives a KEEPALIVE PDU that acknowledges the OPEN PDU sent by the local system, then the local system shall change its state to ESTABLISHED.

— If the local system generates a Stop event, it shall send a CEASE PDU to its peer, and shall then enter the CLOSE-WAIT state.

8.6.3.3 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, maximum number of unacknowledged PDUs that may be outstanding, protocol version number, hold time, and RDI of the other side. In addition, connection may also has been authenticated.

In ESTABLISHED state both sides of a connection may exchange UPDATE PDUs, KEEPALIVE PDUs, IDRP ERROR PDUs, and CEASE PDUs with its peer.

If an OPEN PDU is received from the peer BIS, it shall be ignored: that is, the BIS shall not respond to duplicate delayed OPEN PDUs.

If the local system receives an IDRP ERROR PDU or a CEASE PDU, it shall send the CEASE PDU and shall change its state to CLOSE-WAIT.

If the local system receives a CEASE PDU, it shall change its state to CLOSE-WAIT.

If the local system generates a Stop event, it sends CEASE PDU and changes its state to CLOSE-WAIT.

8.6.3.4 CLOSE-WAIT State

The CLOSE-WAIT state is entered from the ESTABLISHED state, the OPEN-RCVD state, or the OPEN-SENT state.

If a CEASE PDU or an IDRP ERROR PDU is received in the CLOSE-WAIT state, the local system shall immediately change its state to CLOSED.

The local system shall remain in the CLOSE-WAIT state at most **CloseWaitDelayPeriod** seconds. When that time interval has expired, it shall change its state to CLOSED.

If the local system generates a Stop event, it shall enter the CLOSE-WAIT state.

8.6.4 Closing a Connection

The closing of a connection can be initiated by a Stop event generated by the local system, by receipt of an incorrect PDU, or by receipt of a IDRP ERROR PDU or CEASE PDU from the other end of the connection:

— In the case of the Stop event, the local system sends CEASE PDU to the other side of the connection, and then enters the CLOSE-WAIT state.

— In the case of receiving an incorrect PDU, the local system may send an IDRP ERROR PDU, and then enters the CLOSE-WAIT state. Unless otherwise specified, the Data field of the IDRP ERROR PDU shall be empty.

— In the case of receiving an IDRP ERROR PDU, the local system sends CEASE PDU to the other side of the connection, shall deallocate all resources, and shall enter the CLOSED state after expiration of the ″CloseWaitDelayTimer″. Routeing table entries associated with the remote BIS shall be marked as invalid, the local BIS notify its neighbors that they have become infeasible (by sending UPDATE PDUs with the UNREACHABLE attribute), and it shall delete the routes from its Loc-RIBs and its Adj-RIBs-Out. and enters the CLOSE-WAIT state.

— In the case of receiving a CEASE PDU, the local system enters the CLOSE-WAIT state.

While in the CLOSE-WAIT state the local system retransmits unacknowledged IDRP ERROR PDUs. While in the CLOSE-WAIT state the local system shall accept only KEEPALIVE and CEASE PDU′s from the other side of the connection.

When the timeout period specified by managed object **CloseWaitDelayPeriod** expires, the BIS shall enter the CLOSED state, the connection record is deallocated, and the connection cease to exist.

8.7 Version Negotiation

BIS peers may negotiate the version number of IDRP by making multiple attempts to open a BIS-BIS connection, starting with the highest supported version number 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.

8.8 Checksum Algorithm

The checksums used in this international standard shall be generated in accordance with the procedures described in normative annex Appendix B. 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.

8.9 Authentication Mechanism

Note: 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 an appropriate security protocol specified for the provision of secure ISO 8473 communications.

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 shall 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 shall be generated by the following two step process, which is illustrated in Figure 7:

1) An unencrypted checksum that covers the contents of the BISPDU shall be generated by applying the procedures of annex Appendix B 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* shall be encrypted, and the encrypted pattern shall be placed in the Validation Pattern field of the BISPDU.

Notes:

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 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 in 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 shall 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 shall be accepted. If the ″reference pattern″ and the ″received pattern″ are not identical, 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.

8.10 Routeing Information Bases

The Routeing Information Base (RIB) within a BIS consists of three distinct parts, as shown in Figure 8:

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 8.10.1).

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 8.10.1). Information in the Loc-RIB is used to build the Adj-RIBs-Out.

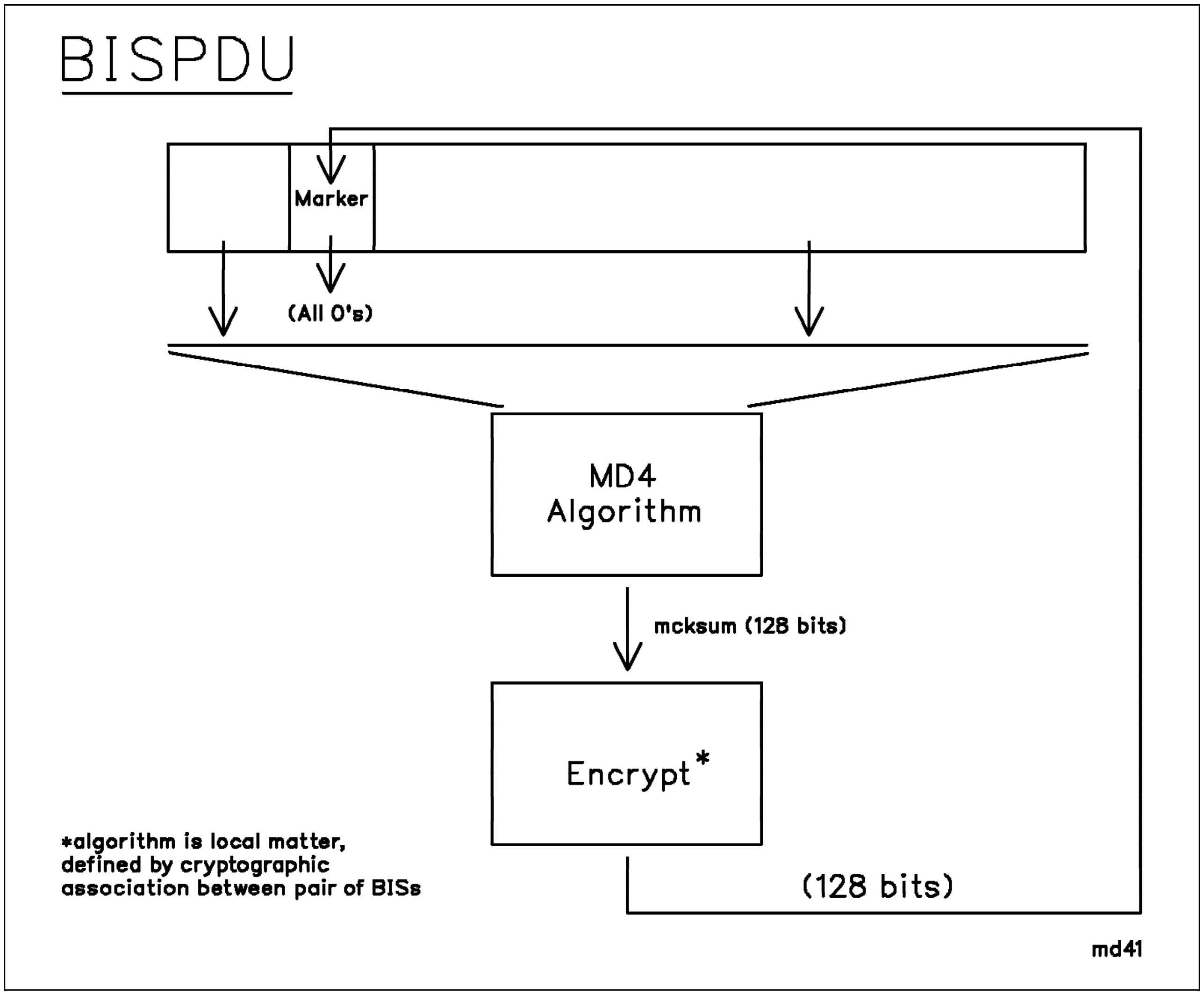
BISPDU
Marker
(All 0's)
MD4 Algorithm
mcksum (128 bits)
Encrypt*
*algorithm is local matter, defined by cryptographic association between pair of BISs
(128 bits)
md41

Figure 7. Validation Pattern in Support of the Authentication Function

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 8.10.1). 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: 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.

8.10.1 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 that form a permissible combination, as defined in clause 8.12.2.

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

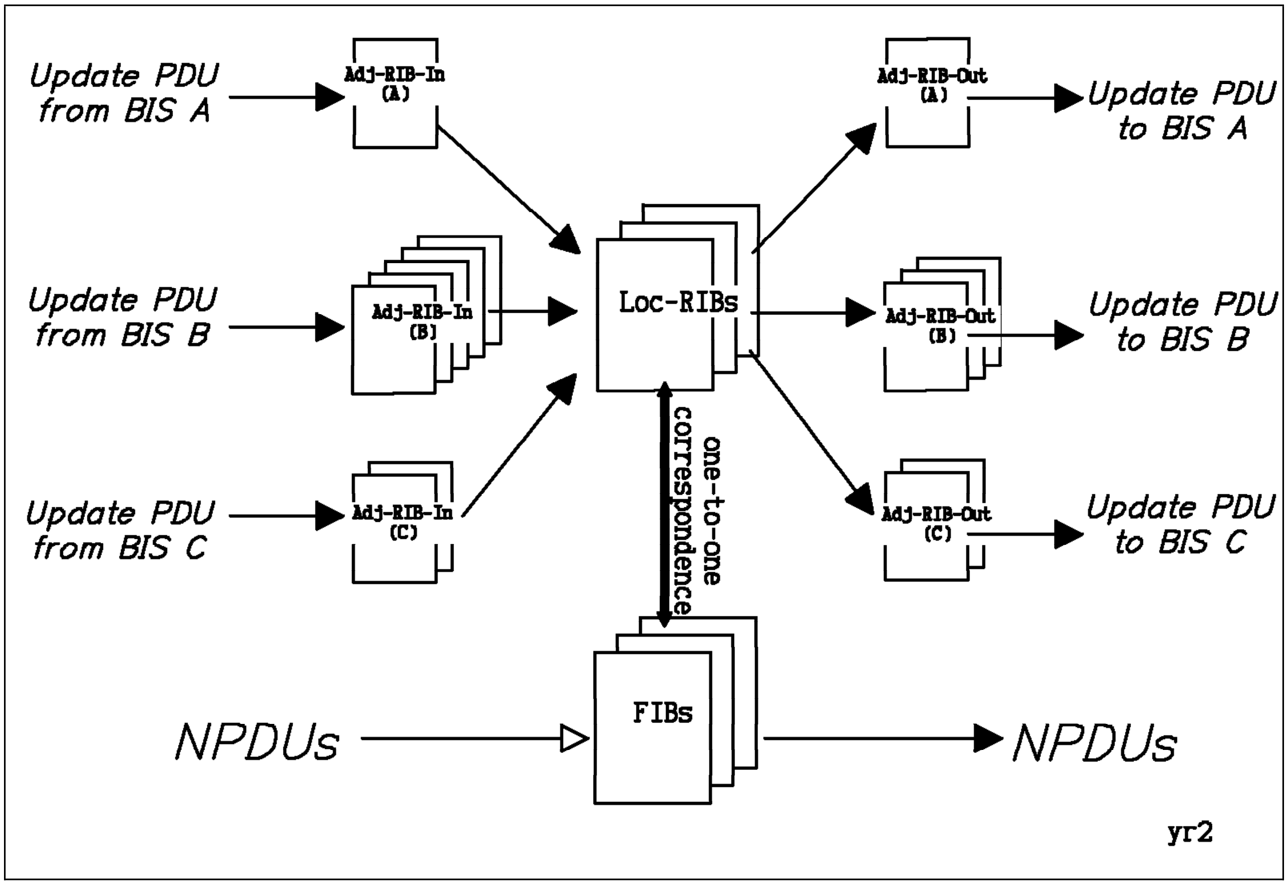

Figure 8. Routeing Information Base. An RIB is comprised of Adj-RIBs-In, Adj-RIBs-Out, and Loc-RIBs

All BISs located within a given routeing domain shall support the same RIB-Atts: that is, managed object **RIB-AttsSet** shall be identical for every BIS within an RD. 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 8.19.2 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: 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.

8.10.2 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: This standard does not prescribe a specific checksum algorithm but notes that the procedures described in Appendix D satisfy the requirements given above. Other approaches can also be used: for example, it may be possible to use an incremental algorithm 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 7.16.3.3.

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.

8.10.3 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:

— *Solicited Refresh*

A BIS may request a neighbor BIS to refresh of 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.

— *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.

8.11 Stability of Routes

Because it does not assume centralized coordination of routeing policies between multiple routeing domains, IDRP provides a method for detecting sets of routeing policies that can not be satisfied simultaneously. If such routeing policies were used for route selection, there could be permanent oscillations in the route selection process.

To detect such problems, each BIS shall observe the following rules:

a) If a BIS determines that a current locally selected route has become unreachable, then the BIS shall advertise this route as unreachable to all of its neighbor BISs before it advertises a replacement route.

This allows a BIS in an adjacent routeing domain to determine if the local BIS has abandoned a previously advertised route because that route has become unreachable or because it has selected a new route which is preferable to the old one. (The old one is still feasible, but is no longer preferred.

b) When it receives a new route in an UPDATE PDU from an adjacent BIS, the local system shall check whether this new route has been chosen by the adjacent RD because the old route become unreachable, or because the adjacent RD prefers it over the route previously advertised by that RD. In the former case, an UPDATE PDU with the UNREACHABLE attribute will have been received for the old route. In the latter case it will not.

   If the adjacent RD has chosen the new route in preference to a previously advertised route, the local BIS shall perform the following actions:

   — If the new route does not cause the local BIS to change any of its currently selected routes, no further checks are needed.

   — If the new route causes the local system to change a currently selected route, the local BIS checks whether the newly advertised route depends on the local RD′s currently selected route. A dependency occurs when the RDI of either the local RD or any confederation to which it belongs is included in the RD_PATH attribute of the newly advertised route. The attribute prefix is then obtained by deleting all path segments of the RD_PATH attribute that occur after (that is, at higher numbered octets) the RDI of the local RD or any confederation to which it belongs.

   If path segments of the attribute prefix do not match the leading path segments of the RD_PATH attribute, then there are inconsistent policies: that is, there is a set of RDs such that a subset of routeing policies of these RDs can not be satisfied simultaneously. The set of these RDs is specified in the RD_PATH attribute of the new route advertised by the adjacent RD.

   Note: The methods described above will detect unsatisfiable policies which have been based on routeing information learned by the methods of this international standard. However, if policies have been based on information learned from other sources, then it may not be possible to detect mutually unsatisfiable polices. That is, detection of mutually unsatisfiable policies may not be possible when some of the routes include the EXT_INFO attribute.

A BIS that conforms to this international standard must support detection of unsatisfiable policies as described above, and it shall notify system management upon detection of a set of unsatisfiable policies. Suppression of such policies is outside the scope of this standard, and is left as a local option.

Note: A particular routeing policy within an RD is called ″active″ when the current route of the RD is selected based on that policy. Otherwise, that policy is said to be ″passive″. The approach outlined above provides a mechanism to detect a set of active routeing policies that can not be satisfied simultaneously; it can not detect sets of passive policies which are mutually unsatisfiable.

A formal proof of the stability of the route selection process is contained in Annex Appendix K.

8.12 Path Attributes

The UPDATE PDU contains fields which carry a set of *path attributes*.

8.12.1 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. Even if a given BIS does not recognize an optional transitive attribute, it may 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 accepted and passed on to other BISs, then the BIS shall set the PARTIAL bit in the attributes flag of its outbound UPDATE PDU to 1

b) If a route with a recognized optional transitive attribute is accepted and passed on to other BISs, then the BIS shall not modify the value of the PARTIAL bit.

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-tranistive attribute.

d) If a route with a recognized optional non-transitive attribute is received, the receiving BIS may accept this information, and may propagate it to other BISs.

Table 1. Path Attribute Characteristics

Attribute	Category	Type Code	Length (octets)	Distinguishing
EXT_INFO	well-known discretionary	1	0	No
RD_PATH	well-known mandatory	2	variable	No
NEXT_HOP	well-known discretionary	3	variable	No
UNREACHABLE	well-known discretionary	4	0	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
LOCAL_PREF	well-known discretionary	8	1	No
TRANSIT DELAY	well-known discretionary	9	2	Yes
RESIDUAL ERROR	well-known discretionary	10	4	Yes
EXPENSE	well-known discretionary	11	2	Yes
SOURCE SPECIFIC QOS	well-known discretionary	12	variable	Yes
DESTINATION SPECIFIC QOS	well known discretionary	13	variable	Yes
HIERARCHICAL RECORDING	well-known discretionary	14	1	No
CO/CL PATH	optional transitive	15	1	No
RD_HOP_COUNT	well-known mandatory	16	1	No
SOURCE SPECIFIC SECU-RITY	well-known discretionary	17	variable	Yes
DESTINATION SPECIFIC SECURITY	well-known discretionary	17	variable	Yes
CAPACITY	well-known discretionary	18	1	Yes
PRIORITY	well-known discretionary	19	1	Yes

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.

The order of appearance of attributes within the path information of a given UPDATE PDU is immaterial; however, the same attribute can appear only once within a given UPDATE PDU. Path attributes are summarized in Table 1; their encoding is described in clause 7.3.

8.12.2 Handling of Distinguishing Attributes

In IDRP certain well-known discretionary path attributes are Distinguishing Attributes (see clause 6.7). They 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 8.13.

For the purpose of constructing multiple routes to a destination Distinguishing Attributes may be grouped into unordered sets, each of which characterizes a particular route. A set may either consist of a single Distinguishing Attribute or multiple Distinguishing Attributes.

IDRP places constraints on the combinations of Distinguishing Attributes that may appear within a single set, namely:

a) A permissible set of Distinguishing Attributes can not include two or more SOURCE SPECIFIC QOS attributes such that the value of the Address Length field or the Address field is different.

b) A permissible set of Distinguishing Attributes can not include two or more DESTINATION SPECIFIC QOS attributes such that the value of the Address Length field or the Address field is different.

c) A permissible set of Distinguishing Attributes can not include any instance of equivalent Distinguishing Attributes (see clause 8.12.3).

All other combinations of Distinguishing Attributes are permitted. A set of Distinguishing Attributes for a single route which is advertised in an UPDATE PDU shall form a permissible combination, as defined above.

8.12.3 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 their type, but require knowledge of both type and value—namely, Source Specific QOS, Destination Specific QOS, Source Specific Security, and Destination Specific Security. 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.

8.13 Path Attribute Usage

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

8.13.1 EXT_INFO

EXT_INFO is a well-known discretionary attribute. It must 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.[8]

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: 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.

8.13.2 RD_PATH

RD_PATH is a well-known mandatory attribute. It shall be present in every UPDATE PDU, and shall be recognized upon receipt by all BISs. The content of this attribute is a list of the RDIs of the RDs and RDCs through which this UPDATE PDU has passed. The components of the list can be expressed as RD_SETs or RD_SEQUENCEs.

When a BIS re-distributes a route which it has learned from a previously received UPDATE PDU, it shall modify the RD_PATH attribute based on the location of the BIS to which the new UPDATE PDU will be sent:

a) When a given BIS sends an UPDATE PDU to another BIS located in its own RD, then the originating BIS shall not modify the RD_PATH attribute associated with the route.

b) When a given BIS sends an UPDATE PDU to a BIS located in an adjacent RD, then the BIS that originates the UPDATE PDU shall update the RD_PATH attribute by attaching the RDI of its own RD to the end of the existing RD_PATH attribute: that is, the newly appended information shall occupy the highest numbered octets of the RD_PATH attribute.

[8] For example, information obtained from the managed object **INTERNAL-SYSTEMS** or information obtained from UPDATE PDUs which do not contain the EXT_INFO attribute are learned by methods within IDRP's scope; however, manually configured reachability information about an RD which does not run IDRP is an example of information which is learned by means outside IDRP's scope.

The BIS that modifies the RD_PATH attribute is free to collapse the previously received RD_PATH information in accordance with its local policies: for example, it may create an RD_SET from a received RD_SEQUENCE.

When a BIS advertises a route whose destinations are located within its own RD, then:

a) the originating BIS shall include its own RDI in the RD_PATH attribute of all UPDATE PDUs sent to BISs located in adjacent RDs. (In this case, the RDI of the originating BIS′s RD will be the only entry in the RD_PATH attribute.)

b) the originating BIS shall include an empty RD_PATH attribute in all UPDATE PDUs sent to BISs located in its own RD. (An empty RD_PATH attribute is one whose length field contains the value zero.)

If the originating BIS also is a member of a Routeing Domain Confederation, then it shall additionally comply with the requirements of clause 8.14: in particular, RDs which are members of a confederation shall not be permitted to collapse RD_SEQUENCE information into an RD_SET when the original sequence contains an entry marker (see clause 8.14.4), nor shall they be permitted to generate information expressed as an RD_SET.

8.13.3 NEXT_HOP

NEXT_HOP is a well-known discretionary attribute. It must 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 or IS 10030) to ascertain if there is direct reachability between them. Examples of such media are IEEE 802.2, SMDS, and X.25.

Consider three BISs attached to a fully connected transitive subnetwork, as shown in Figure 9: 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. Let us further assume that C propagates an UPDATE PDU to B. 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 its own SNPAs 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 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 its own SNPAs 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 place.

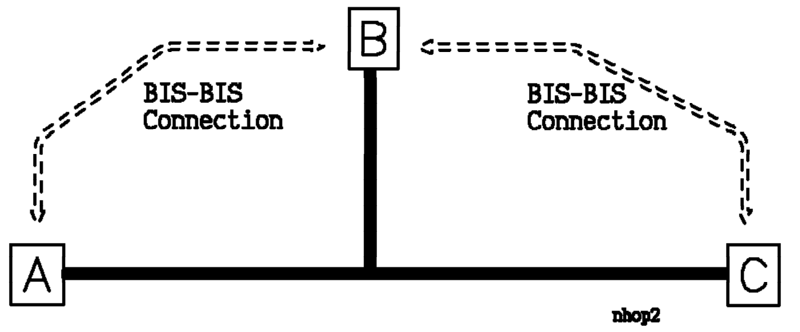

Figure 9. A Transitive Fully Connected Subnetwork

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.)

Notes:

a) The rules stated above 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 8.13.2.

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.

8.13.4 UNREACHABLE

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

A path to a given set of destinations is called *feasible* if it was advertised in an UPDATE PDU that did not contain the UNREACHABLE attribute; the path becomes *unfeasible* when it is advertised in an UPDATE PDU that contains the UNREACHABLE attribute.

If a given BIS receives an UPDATE PDU that contains an UNREACHABLE attribute, then the receiving BIS shall delete the corresponding route from its Adj-RIB-In. If the corresponding route is present in its Loc-RIB, the BIS shall also re-run its Decision Process, using the newly updated contents of its Adj-RIBs-In, and shall update its Adj-RIBs-Out and FIBs accordingly.

A given route may become unavailable for further use either because it was advertised to the local BIS with an UNREACHABLE attribute, or the local policies of the BIS have declared it to be so. For example, a local BIS may choose to further restrict distribution of a previously advertised feasible route, thus making it unavailable to some of the BISs who had previously been permitted to use it.

When the Decision Process of a BIS makes a previously advertised route unavailable for further use, then the BIS shall remove that route from all Adj-RIBs-Out in which it was contained, and shall advertise it with an UNREACHABLE attribute to all neighbor BIS to whom it had previously advertised the corresponding feasible route.

Since the Adj-RIB-In into which a given route will be stored can be unambiguously determined by the distinguishing attributes of the UPDATE PDU in which it is advertised, a BIS shall advertise an unfeasible route by generating an UPDATE PDU that contains the following minimum amount of information: the UNREACHABLE attribute, the distinguishing attributes of the previously feasible route, and the NLRI of the previously feasible route. It is not required that the non-distinguishing attributes of the previously feasible route be included.

8.13.5 DIST_LIST_INCL

DIST_LIST_INCL is a well-known discretionary attribute. It must 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 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. 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.

8.13.5.1 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 8.13.14, shall take precedence over those imposed by DIST_LIST_INCL.

8.13.6 DIST_LIST_EXCL

DIST_LIST_EXCL is a well-known discretionary attribute. It must 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. 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. 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 contained 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.

8.13.6.1 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 in clause 8.13.6, shall take precedence over those imposed by the HIERARCHICAL_RECORDING attribute.

8.13.7 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 re-distribute the attribute to any BISs which are not located within its own RD.

8.13.8 LOC_PREF

LOCAL_PREF is a well-known discretionary attribute that must be included in all UPDATE PDUs that a given BIS sends to the other BISs located in its own routeing domain. Its principal use is to permit the detection of inconsistent routeing decisions among a set of BISs that are all located in the same routeing domain, as described in clause 8.15.2.

A BIS shall not include this attribute in UPDATE PDUs that it sends to BISs located in an adjacent routeing domain. If it is contained in an UPDATE PDU that is received from a BIS which is not located in the same routeing domain as the receiving BIS, then this attribute shall be ignored by the receiving BIS.

8.13.9 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 500 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.

8.13.10 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:

$$K \times (1 - \left[(1 - RRE/K) \times (1 - RDLRE/K)\right])$$

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.

8.13.11 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 Residual Error attribute shall not be modified, but shall be distributed with the same value that was present in the received UPDATE PDU.

8.13.12 SOURCE SPECIFIC QOS

SOURCE SPECIFIC QOS is an well-known discretionary attribute that allows a BIS to control the ability of ISO 8473 NPDUs with a given source NSAP address to transit through its local RD. The finest granularity of this control is a single End system. A BIS shall include this attribute in its UPDATE PDU to indicate that it supports source specific routeing based on the reported QOS value, and that it maintains Adj-RIBs and a Loc-RIB distinguished by the indicated source specific QOS value.

In accordance with clause 7.5.6.1 of ISO 8473, the value of the QOS Value field of the SOURCE SPECIFIC QOS field shall be unique and unambiguous in the context of the QOS maintenance system employed by the authority responsible for assigning the source NSAP address.

Note: Methods for the assignment of unambiguous and unique QOS Values throughout the global OSIE are outside the scope of IDRP.

This attribute may be originated only by those BISs that reside in the same RD as the destination End system(s) reported in the NLRI of the UPDATE PDU.

When a BIS re-distributes a route which has been learned in an UPDATE PDU that contains the Source Specific QOS attribute, the new UPDATE PDU shall contain the Source Specific QOS attribute which shall have the same QOS value and address fields that are associated with the route, and the metric field (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 are not within the scope of IDRP.

8.13.13 DESTINATION SPECIFIC QOS

DESTINATION SPECIFIC QOS is an well-known discretionary attribute that allows a BIS to control the ability of ISO 8473 NPDUs with a given destination NSAP address to transit through its local RD. The finest granularity of this control is a single End system. A BIS shall include this attribute in its UPDATE PDU to indicate that it supports destination specific routeing based on the reported QOS value, and that it maintains Adj-RIBs and a Loc-RIB distinguished by the indicated destination specific QOS value.

In accordance with clause 7.5.6.2 of ISO 8473, the value of the QOS Value field of the DESTINATION SPECIFIC QOS field shall be unique and unambiguous in the context of the QOS maintenance system employed by the authority responsible for assigning the destination NSAP address.

Note: Methods for the assignment of unambiguous and unique QOS Values throughout the global OSIE are outside the scope of IDRP.

This attribute may be originated only by those BISs that reside in the same RD as the destination End system(s) reported in the NLRI of the UPDATE PDU.

When a BIS re-distributes a route which has been learned in an UPDATE PDU that contains the Destination Specific QOS attribute, the new UPDATE PDU shall contain the Destination Specific QOS attribute which shall have the same QOS value and address fields that are associated with the route, and the metric field (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 are not within the scope of IDRP.

8.13.14 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 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.

When announcing unfeasible routes, the HIERARCHICAL_RECORDING attribute need not be included; however, propagation of the unfeasible routes is governed by the rules that were used when the corresponding feasible route was originally propagated. If present in an UPDATE PDU that advertises an unfeasible route, the HIERARCHICAL_RECORDING attribute shall be ignored.

8.13.14.1 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 in clause 8.13.6, 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 8.13.14, shall take precedence over those imposed by DIST_LIST_INCL.

8.13.15 CO/CL PATH

CO/CL PATH is an optional transitive attribute which is 1-octet in length. The usage rules are:

a) A BIS that originates an UPDATE PDU may optionally include this attribute. If present, the source BIS shall set bit 1 to indicate if the path should be connection-mode or connectionless-mode.

b) If this attribute was not present in the inbound UPDATE PDU, then it shall not be included in the outbound PDU.

c) If a BIS sends an UPDATE PDU containing the CO/CL attribute to an IFU, it shall set bit 8 of the attribute to 1.

d) If a BIS chooses to advertise a route learned from an UPDATE PDU that contained the CO/CL attribute, its outbound UPDATE PDU shall also contain the CO/CL path attribute. Bit 1 of the attribute shall remain at the same value it had in the UPDATE PDU that advertised the route. If the outbound UPDATE PDU is not sent to an IFU, then bit 8 shall also remain at the same value it had in the UPDATE PDU that advertised the route.

8.13.16 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: 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.

8.13.17 SOURCE SPECIFIC SECURITY

SOURCE SPECIFIC SECURITY is a well-known discretionary attribute that allows a BIS to specify the security level that is associated with a given path, and to limit usage of that path to systems having the NSAP address prefixes listed in this attribute. The finest granularity of this control is a single End system. A BIS shall include this attribute in its UPDATE PDU to indicate that it supports the source specific security level reported in the security level field, and that it maintains Adj-RIBs and a Loc-RIB distinguished by the indicated source specific security level.

In accordance with clause 7.5.3.1 of ISO 8473, the value of the security level field shall be unique and unambiguous in the context of the security classification system employed by the authority responsible for assigning the source NSAP address.

Note: Methods for the assignment of unambiguous and unique security levels throughout the global OSIE are outside the scope of IDRP.

This attribute may be originated only by those BISs that reside in the same RD as the destination End system(s) reported in the NLRI of the UPDATE PDU.

When a BIS re-distributes a route which has been learned in an UPDATE PDU that contains the Source Specific Security attribute, the new UPDATE PDU shall contain the Source Specific Security attribute which shall have the same security level and address fields that are associated with the route.

8.13.18 DESTINATION SPECIFIC SECURITY

DESTINATION SPECIFIC SECURITY is a well-known discretionary attribute that allows a BIS to specify the security level that is associated with a given path, and to limit usage of that path to systems having the NSAP address prefixes listed in this attribute. The finest granularity of this control is a single End system. A BIS shall include this attribute in its UPDATE PDU to indicate that it supports the destination specific security level reported in the security level field, and that it maintains Adj-RIBs and a Loc-RIB distinguished by the indicated destination specific security level.

In accordance with clause 7.5.3.2 of ISO 8473, the value of the security level field shall be unique and unambiguous in the context of the security classification system employed by the authority responsible for assigning the destination NSAP address.

Note: Methods for the assignment of unambiguous and unique security levels throughout the global OSIE are outside the scope of IDRP.

This attribute may be originated only by those BISs that reside in the same RD as the destination End system(s) reported in the NLRI of the UPDATE PDU.

When a BIS re-distributes a route which has been learned in an UPDATE PDU that contains the Destination Specific Security attribute, the new UPDATE PDU shall contain the Destination Specific Security attribute which shall have the same security level and address fields that are associated with the route.

8.13.19 CAPACITY

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

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 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 **Capacity**.

If a BIS redistributes a route that was advertised with the CAPACITY attribute present, it shall include the CAPACITY attribute in its outbound UPDATE PDU, and shall set its value equal to 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**.

8.13.20 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 lower 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**.

8.14 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 exactly like a single routeing domain: for example, it has an identifier which is an RDI. Other RDs can develop policies with respect 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 4.7).

8.14.1 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 8.15.2 will detect both internal and external routeing inconsistencies, and the methods of clause 8.11 will detect mutually unsatisfiable policies between itself and BISs located in other routeing domains.

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. Therefore, routes between two RDs in the same confederation will be constrained to remain within that confederation. Any attempt to construct a route which leaves the confederation and then later re-enters it will be detected as a loop, and such routes will be rejected.

8.14.2 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 8.3.

8.14.3 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 comparison 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.

8.14.4 Entry Marker for an RDC

All UPDATE PDUs contain the RD_PATH attribute, which reports the RDIs of the routeing domains and confederations through which the UPDATE PDU has travelled. To routeing domains are outside a given confederation, the RDI of a confederation is indistinguishable from the RDI of a routeing domain. However, a marker is needed for the benefit of the members of a given RDC to denote that the UPDATE PDU has either entered or exited the confederation. Knowledge of entry or exit is only needed by confederation members, and therefore the marker for a given confederation shall only be present in the

RD_PATH attribute while the UPDATE PDU is travelling within that confederation.

The Entry Marker shall be encoded in the RD_PATH attribute as an RDI of length 0. To denote that an UPDATE PDU has entered one or more confederations, the receiving BIS includes in its outbound UPDATE PDUs an Entry Marker followed by the RDI(s) of the entered confederation(s).

Note: This use of a zero-length RDI does not conflict with the other use described in clause 8.13.2 When used as an Entry Marker, the zero-length RDI will be followed by one or more non-zero length RDIs. When the zero-length RDI is used according to clause 8.13.2, it will constitute the entire contents of the RD_PATH attribute.

8.14.5 Generating RD_PATH Information

When a BIS in a given RDC generates information about destinations contained within itself, it shall examine the information contained in its managed object **RDC-Config** in order to determine the ordering relationship(s) that exist among all the RDCs to which it belongs. It shall then construct an RD_PATH attribute in the form of an RD_SEQUENCE, as follows:

a) For each RDC to which it belongs, the BIS shall attach the 2-tuple <Entry Marker><RDI of the RDC>.

b) If a confederation, RDC-A, is nested within another confederation, RDC-B, then the 2-tuple for RDC-B shall precede the 2-tuple for RDC-A.

c) The 2-tuples of overlapping confederations may be listed in any order, as long as the order implied by any nesting relationships is maintained.

d) After the 2-tuples of all its confederations are listed, the BIS shall append the RDI of its own routeing domain.

8.14.6 Updating Received RD_PATH Information

When a BIS in a given RDC receives an UPDATE PDU from another BIS, it shall update the RD_PATH attribute of any UPDATE PDUs that it subsequently transmits according to the following rules:

a) If the UPDATE PDU was received from a BIS in an adjacent RD and is being advertised to a BIS located in the RD of the advertising BIS, then the RD_PATH attribute of the advertised UPDATE PDU shall be updated by adding the following 2-tuple for each RDC that the inbound UPDATE PDU has entered: <Entry Marker><RDI of entered RDC>.

   The 2-tuple of each RDC that is entered shall be listed according to local partial order that prevails among them, as contained in managed object **RDC-Config**. That is, if RDC-A is nested within RDC-B, then the 2-tuple for RDC-B shall precede the 2-tuple of RDC-A. The 2-tuples of overlapping RDCs may be listed in any order, as long reporting order implied by any nesting relationships is maintained.

b) If the UPDATE PDU was received from a BIS in its own RD and is being advertised to a BIS in an adjacent RD, then the BIS shall use the methods of clause 8.14.3 to establish which confederations the new UPDATE PDU will exit. For each such confederation, the advertising BIS shall scan from the back of the previous RD_PATH attribute, looking for the occurrence of the 2-tuple corresponding to the RDC that is being exited. The BIS shall apply the following procedures to each of the exited confederations. The procedures shall be applied first to any exited confederation that is located at the lowest level of the local partial order, as described in managed object **RDC-Config**. The procedures shall then be applied iteratively to the other exited confederations, always working from those at the lowest level to those at the higher levels.

   For a given exited confederation (for example, the confederation whose RDI is ″X″), the advertising BIS shall then update the RD_PATH in the PDU to be advertised, as follows:

   1) If the 2-tuple <Entry Marker><X> is not found, the route is in error, and the BIS shall issue a IDRP ERROR PDU which reports a Misconfigured_RDCs error.

   2) If the 2-tuple is found and there are no other intervening 2-tuples, then the RD_PATH from the 2-tuple to the end of the attribute shall be replaced by the RDI of the exited confederation.

   3) If an intervening 2-tuple (for example, <Entry Marker><RDC-Y>) is found, and RDC-X is known to be nested within RDC-Y, then:

      — if there are no RDIs listed between the 2-tuple for RDC-X and the 2-tuple for RDC-Y, then RDC-Y and RDC-X were entered simultaneously. In this case, the contents of the RD_PATH attribute from the end back to the 2-tuple for RDC-X shall be replaced by the sequence <Entry Marker><RDC-Y>.

      — if there are any RDIs listed between the 2-tuple for RDC-X and the the 2-tuple for RDC-Y, this indicates that RDC-X is not in fact nested within RDC-Y, and the route is in error. The local BIS shall send an IDRP_ERROR PDU that reports a Misconfigured_RDCs error.

   4) If a 2-tuple for RDC-Z is found, but the local BIS′s routeing domain is not a member of RDC-Z, then the route is in error, and the

local BIS shall send a IDRP ERROR PDU that reports a Misconfigured_RDCs error.

5) If a 2-tuple for RDC-Z is found, where RDC-X and RDC-Z are overlapping confederations, the the portion of the route between RDC-X and RDC-Z shall be replaced by the sequence <RDC-X><Entry Marker><RDC-Z>.

c) If the UPDATE PDU was received from a BIS in an adjacent RD and is being advertised to a BIS in an adjacent RD, then the local BIS shall update the RD_PATH attribute by appending the 2-tuples of each confederation of which it is a member, according to the partial order that prevails among them. That is, if RDC-A is nested within RDC-B, then the 2-tuple for RDC-B shall precede the 2-tuple of RDC-A. The 2-tuples of overlapping RDCs may be listed in any order, as long reporting order implied by any nesting relationships is maintained.

8.15 Update-Receive Process

The Update-Receive process accepts incoming BISPDUs as its input and updates the appropriate Adj-RIBs-In, based on their content.

— When an UPDATE PDU is received which advertises a new or a changed route, the BIS shall select the Adj-RIB-In whose RIB-Att matches distinguishing path attributes reported in the UPDATE PDU. The new routeing information shall then be entered into the appropriate Adj-RIB.

— If the newly received route is marked as unreachable, and a previous (feasible) route is present in the BIS's Loc-RIB, Adj-RIBs-Out, and FIBs, the BIS shall remove the old route from its Loc-RIB, Adj-RIBs-Out, and FIBs, and shall transmit an UPDATE PDU to those adjacent BISs to whom it had advertised the previously feasible route, announcing that the route is now unfeasible.

— If an UPDATE PDU is received whose distinguishing attributes do not match any of the local BIS's RIB-Atts, the UPDATE PDU shall be discarded, and no further processing shall be performed on it.

— If an IDRP ERROR PDU, CHECKSUM PDU, CEASE PDU, or KEEP_ALIVE PDU is received, the BIS shall take the actions dictated by its Finite State Machine, as described in clause 8.6.1.

8.15.1 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. Two types of routeing information inconsistency may occur: internal and external.

— 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.

— An external inconsistency can occur when a pair of routeing domains is directly connected by more than a single link. Even if there is no internal inconsistency and all the BIS within the domain select the same path, different BISs may make different decisions about the reachability information that will be propagated on external links, even when propagating it to the same adjacent routeing domain. Propagation of inconsistent routeing information outside of a routeing domain results in an external inconsistency.

Thus, internal consistency is a necessary (but not sufficient) condition for the external consistency.

8.15.2 Detecting Inconsistencies

The Update-Receive and Update-Send processes of each BIS shall support the LOCAL_PREF Attribute, which is a well-known discretionary attribute. Its value for a particular route is determined by the degree of preference associated with that route as computed by the local BIS.

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

A BIS shall detect inconsistent routeing decisions (and, therefore, internal inconsistencies) as follows:

a) A route received from a BIS located in an adjacent routeing domain is evaluated by the receiving BIS, which assigns it a degree of preference. The newly received route will be propagated to all other BISs in the routeing domain by means of an UPDATE PDU with the LOCAL_PREF set equal to the locally computed degree of preference if either 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 had assigned to other routes —with both the same destinations and the same Distinguishing Attributes—that have been received from BISs located in adjacent routeing domains, or

2) there are no other routes —with both the same destinations and the same Distinguishing Attributes—that have been received

from BISs located in adjacent routeing domains

b) A BIS shall detect internal inconsistencies by evaluating the route carried in an UPDATE PDU received from BISs located in its own routeing domain, and assigning it a degree of preference. This calculated degree of preference is compared to the value of the LOCAL_PREF Attribute contained in the incoming UPDATE PDU. If these values differ, the routeing policies of the two BISs have internal inconsistencies; the BIS receiving the inconsistent information shall report the inconsistency to system management.

Note: If, for a given value of NLRI, all UPDATE PDUs received from BISs located in a single adjacent routeing domain do not contain identical path attributes, except for NEXT_HOP and LOCAL_PREF, then there is an external inconsistency. Such inconsistencies may be permanent (for example, because of a flawed decision process) or transient (for example, the UPDATE PDUs from all BISs in an adjacent RD can not all be received simultaneously). The detection of, and response to, external inconsistencies is left as a local policy matter.

8.16 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 8.16.2; rules for information exchange between BIS located in the same domain are given in clause 8.16.1.

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: 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.

8.16.1 Internal Updates

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 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 either 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 over inter-domain links, or

   2) there are no other routes—with the same destinations and the same Distinguishing Attributes—that have been received over inter-domain links

c) When a BIS receives a report of an unfeasible route from a BIS in an adjacent RD, it shall advertise that route to all other BISs in its routeing domain, indicating that it has become unreachable, if the following condition occurs:

   1) a corresponding feasible route to the same destination has the highest degree of preference among all routes to that destination which have been received over inter-domain links (that is, the local BIS had been advertising the route as feasible before it received the UPDATE PDU which reported it as unreachable.)

8.16.2 External Updates

Whenever a BIS selects a new route or determines that the reachable destinations within its own RD have changed, it shall generate an UPDATE PDU and forward it to each of its neighbor BISs in adjacent routeing domains in accordance with its own local policies for information distribution. However, the UPDATE PDUs shall not be distributed outside the RD until the internal update procedure of clause 8.16.1 has been completed.

To insure that consistent information will be propagated externally by the routeing domain, no BIS shall propagate an UPDATE PDU to a BIS in an adjacent routeing domain until it has propagated that information internally and has received an acknowledgement from all BISs in its own routeing domain with which it has an established BIS-BIS connection.

Furthermore, 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 8.19.3.

8.16.3 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.

8.16.3.1 Frequency of Route Selection

The architectural constant **MinRouteSelectionInterval** determines the minimum amount of time that must elapse between advertisements of better routes received by a given BIS from systems located in other routeing domains.

Since fast convergence is needed within an RD, this constant does not apply to advertisement of better routes chosen as a result of updates from BISs located in the advertising BIS′s own RD. To avoid long-lived black holes, it does not apply to advertisement of previously selected routes which have become unreachable. In both of these situations, the local BIS must advertise such routes immediately.

If a BIS has selected new routes based on updates from BISs in adjacent RDs, but have not yet advertised them because the **MinRouteSelectionInterval** has not yet elapsed, the reception of any routes from other BISs in its own RD shall force the **MinRouteSelectionInterval** timer to expire, and shall trigger a new selection process that will be based on both updates from BISs in the same RD and in adjacent RDs.

8.16.3.2 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.

8.16.3.3 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 **MinRouteSelectionInterval** 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 J, 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 Appendix C, using the architectural constant **jitter**.

8.17 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, as illustrated in Figure 4. 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 Adj-RIBs-Out.

The selection process is formalized by defining a function that takes the attributes of a given path as an argument and returns a non-negative integer denoting the degree of preference for the path. Path selection then consists of application of this function to all feasible paths to a given destination, followed by the choice of the one with the highest degree of preference. The criteria for assigning the degree preference to a path are determined locally by each RD and will be reflected in its PIB.

If the degree of preference of a route currently installed in the Loc-RIB is less than the degree of preference assigned to a new route, then the current route in the Loc-RIB shall be replaced with the new route, and the new route shall be propagated to the adjacent BISs (subject to the DISTRIBUTION_LIST attribute constraints).

8.17.1 Breaking Ties

If there are several routes to the same destination which all have the same degree of preference, the decision process shall choose a single route from among them. If the local BIS has a locally-generated routes (to the same destination and with the same degree of preference), it shall be included in the following selection process, along with the routes advertised by other BISs:

a) Identify the route advertised by the BIS whose binary encoded NET has the lowest value when it is considered to be an unsigned binary integer.

b) .Assemble a set of routes whose path attributes, except for NEXT_HOP, match the path attributes of the route identified in step a

c) Separate the set in step b into two groups: group 1 contains routes advertised by BISs located outside the local routeing domain, and group 2 contains routes advertised by BISs located within the routeing domain.
    1) If group 1 is not empty, select the route from this group that was advertised by the BIS with the numerically lowest NET.
    2) If group 1 is empty, select the route from group 2 that was advertised by the BIS with the numerically lowest NET.

8.17.2 Updating the Loc-RIBs

When the Decision Process operates on information stored in one of its Adj-RIBs-In, it shall identify the single Loc-RIBs that may potentially be updated with route information in that PDU. The Loc-RIB shall be the one whose RIB-Att is identical to the set of Distinguishing Attributes contained in the route being processed. If the BIS does not have a Loc-RIB with an identical RIB-Att, then the BIS shall not install this route in its Loc-RIBs; hence, it can not advertise this route to other BISs.

Because a route with a given set of Distinguishing Attributes can not be propagated by a BIS which does not support that set of Distinguishing Attributes, all BISs that advertise this route form a contiguous network (region). Since a set of Distinguishing Attributes maps into a specific RIB-Att, it follows that a RIB-Att defines a subset of the physical network that contains routes with a given set of Distinguishing Attributes.

The process of comparing RIB-Atts of various Loc-RIB with Distinguishing Attributes of a route stored in an Adj-RIB and the subsequent selection of a Loc-RIB forms the core of the pa2riba() procedure that is used by the example route selection algorithm described in Appendix M.

8.17.3 Path Selection

IDRP does not rely on the existence of a universally agreed-upon metric to exist between multiple RDs. Therefore, IDRP allows each RD to apply its own set of criteria for path selection, as determined by its local PIB.

The path selection process takes place in two stages. First, when a BIS receives path information from a BIS in an adjacent RD, that BIS selects the best route to each destination for each supported RIB-Att, and it redistributes those routes to all other BISs located in its own RD. Second, each BIS selects the best route from all those distributed within the RD and announces its choice to adjacent RDs (subject to local policy constraints). This two step process limits the volume of information that must be distributed within an RD, and it insures rapid convergence on new routes.

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 Appendix G.

8.17.4 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 8.17.5. 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.

8.17.5 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.

8.17.5.1 Route Aggregation

Several routes shall not be aggregated into a single route unless the Distinguishing Attributes of each of these route are the same. In particular,

— Feasible routes may be aggregated with feasible routes.

— Unfeasible routes may be aggregated with unfeasible routes.

— Feasible and unfeasible routes can not be aggregated together.

— 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, LOC_PREF, and NEXT_HOP.

8.17.5.2 Aggregating NLRI

The aggregation of the NLRI fields from several routes is straightforward: if a shorter NSAP address prefix can be used to represent the systems (and only those systems) listed in several individual NSAP address prefixes, this may be done. However, a shorter NSAP prefix which would be associated with more systems than were associated with the individual prefixes being aggregated shall not be used.

8.17.5.3 Aggregating Path Attributes

Path attributes that have different type codes can not be be aggregated together. Path attributes of the same type code may be aggregated, according to the following rules:

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: If routes to be aggregated have identical RD_PATH attributes, then the aggregated route has the same RD_PATH attribute as each individual route.

If a route has an RD_PATH attribute that contains a confederation entry marker (see 8.14.4), then that route shall not be aggregated with any other route.

If routes to be aggregated have different RD_PATH attributes, then the rules for determining the RD_PATH attribute of the aggregated route are specified by the following algorithm:

a) Set RD_PATH of the aggregated route to empty.

b) Determine the longest common initial sequence of the RD_PATH segments among all the routes to be aggregated.

c) Append this longest common initial sequence (as is) to the RD_PATH of the aggregated route.

d) Delete this longest common initial sequence from the RD_PATH attribute of each of the routes to be aggregated.

e) Determine the longest different initial sequences of the RD_PATH segments among all the routes to be aggregated.

f) Combine all RDI's in these sequences into an RD_SET path segment and append it to the RD_PATH attribute of the aggregated route.

g) Delete sequences determined in step e from the RD_PATH attribute of each of the routes to be aggregated.

h) Repeat steps b to g until there is at least one route with non-empty RD_PATH attribute.

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.

Source Specific QOS attributes: The rules for determining the value of the Source Specific QOS attribute of the aggregated route are determined by the authority responsible for assigning the Source Specific QOS.

Destination Specific QOS attributes: The rules for determining the value of the Destination Specific QOS attribute of the aggregated route are determined by the authority responsible for assigning the Destination Specific QOS.

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 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.

Source Specific Security attribute: The rules for determining which source specific security levels can be aggregated, and the rules for doing so, are determined by the authority responsible for originating the attribute.

Destination Specific Security attributes: The rules for determining which destination specific security levels can be aggregated, and the rules for doing so, are determined by the authority responsible for originating the attribute.

Priority The value of the PRIORITY attribute of the aggregated route shall be equal to the minimum value of the PRIORITY fields in the routes being aggregated. A route that does not contain the PRIORITY attribute shall be treated as if it contained a priority value of 0.

Capacity The value of the CAPACITY attribute of the aggregated route shall be equal to the minimum value of the CAPACITY fields in the routes being aggregated.

8.17.6 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.

8.18 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.

8.18.1 Requirements on ISO 8473 Receive Processing

The ISO 8473 protocol machine shall process inbound NPDUs as follows:

a) NPDUs addressed to the local BIS shall be processed according to the appropriate ISO 8473 functions, for example:

— Header Format Analysis
— PDU Lifetime Control
— Reassembly
— Discard PDU
— Error Reporting
— PDU Header Error Detection
— Priority

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 protocol machine for further processing by its Update-

Receive Process (clause 8.15) and error handling processes (clause 8.19), as appropriate.

However, if the source address of the NPDU identifies a system that is not listed in these managed objects, then the NPDU shall be rejected by the receiving BIS, and the BIS shall send a notification (″packetbomb″) to system management.

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 9.4, item b2.)

b) NPDUs addressed to systems other than the local BIS shall be processed according to the appropriate ISO 8473 functions, for example:

- Header Format Analysis
- PDU Lifetime Control
- Discard PDU
- Error Reporting
- PDU Header Error Detection
- Complete Source Routeing
- Record Route
- QOS Maintenance
- Priority
- Security
- Congestion Notification

Then, the 8473 protocol shall hand the NPDU over to the IDRP CLNS forwarding process (see clause 9), which will then forward the NPDU to a next-hop which has been determined by the methods of this international standard. In particular, the ISO 8473 protocol machine shall not apply its ″Route PDU function″ or its ″Forward PDU function″ to the NPDU, unless the NPDU contains the ISO 8473 Complete Source Route parameter.

8.19 Error Handling for BISPDUs

This section describes actions to be taken when errors are detected while processing BISPDUs.

8.19.1 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:

- Length field of a PDU header less than 31 octets or greater than the Segment Size specified by the remote system′s OPEN PDU,
- Length field of an OPEN PDU less than minimum length of an OPEN_ PDU
- Length field of an UPDATE PDU less than minimum length of an UPDATE PDU
- Length field of KEEPALIVE PDU not equal to 31
- The BIS-BIS connection was established using authentication code 1 (checksum without auhentication) and the validation pattern in the BISPDU header does not match the locally computed pattern
- Type field in the BISPDU is not recognized

8.19.2 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.

- 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).
- 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.
- If the Outstanding PDUs field of the OPEN PDU is zero, the error subcode of the IDRP ERROR PDU is set to Bad_Outstanding_PDUs. The Data field of the IDRP ERROR PDU is a 1 octet unsigned integer which contains the erroneous Outstanding PDUs field.
- 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).
- 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.
- 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.

— 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 NOTIFICATION shall report the offending Confed-IDs field from the rejected OPEN PDU.

If the following condition occurs, it shall be reported to system management, and normal processing of the OPEN PDU shall continue:

— A given BIS shall issue a report to system management when all of the following conditions are met simultaneously:

  a) The managed object **RDC-Config** for the receiving BIS indicates that it is a member of several RDCs with a given nesting relationship
  b) The incoming OPEN PDU reports that the remote BIS is a member of some but not all of the same RDCs.

Note: These mismatches may be a transient condition which will only exist during the formation or deletion of an RDC; hence, the local BIS will continue with normal processing of the OPEN PDU, but shall be required to notify network management of the situation.

8.19.3 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.

— 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.

— 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.

— 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.

— 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.

— 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.

— 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.

— 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 incorrect attribute (type, length and value). No further processing shall be done, and all information in the UPDATE PDU shall be discarded.

— If the non-empty set of distinguishing attributes of 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.

— 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.

— 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.

— 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: It is permissible for an UPDATE PDU to contain neither the DIST_LIST_INCL nor the DIST_LIST_EXCL attributes. According to clause 8.13.5, the absence of both the DIST_LIST_INCL and DIST_LIST_EXCL attributes implies that all RDs and all RDCs may process the routeing information.

— 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.

— If RDCs are supported and any of the error conditions noted in clause 8.14.6 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.

8.19.4 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 8.6.4.

Note: 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.

8.19.5 Hold Timer Expired Error Handling

If a system does not receive successive KEEPALIVE, UPDATE or CHECKSUM PDU 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.

9. 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 9.1.

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 9.2.

    2) It shall next apply the procedures of clause 8.10.1 to determine if the NPDU-derived Attributes match any of the RIB-Atts of the information base(s) supported by the local BIS:

        i) If there is no match, then the local BIS shall perform the ISO 8473 ″Discard PDU Function″ (see clause 6.9 of ISO 8473), and it shall generate an ER PDU with the parameter value set to ″Unsupported Option not Specified″.

        ii) If there is a match, then the local BIS shall proceed as described in clause 9.4.

9.1 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.

9.2 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 high order bits of the 8473 security parameter are decoded. If they equal to ′01′, then the security parameter corresponds to the SOURCE SPECIFIC SECURITY path attribute; if equal to ′10′, it corresponds to the DESTINATION SPECIFIC SECURITY path attribute.

— The remainder of the NPDU-Derived Distinguishing Attributes are derived by decoding the first octet of the QOS maintenance parameter, as shown in Table 2

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.

9.3 Matching RIB-Att to NPDU-derived Distinguishing Attributes

Within the BIS, each of its FIB(s) has an unambiguous RIB-Att (see clause 8.10.1) 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:

   — Capacity

   — Transit delay

   — Residual error

   — Expense

   — Priority

c) Each instance of a type-value specific attribute in the NPDU-derived Distinguishing Attributes must have a corresponding instance in the FIB-Att with the same type and an equivalent value. The type-value specific attributes supported by IDRP are:

   — Source Specific QOS

   — Destination Specific QOS

   — Source Specific Security

   — Destination Specific Security

   For the source and destination specific attributes, the NPDU-derived Distinguishing Attribute will report only a single relevant address, while in general the corresponding FIB-Att attribute apply to multiple addresses. A match occurs when the attribute types are identical and the address carried in the 8473 parameter is a subset of the address information contained in the FIB-Att. In searching for a match, the address information in the FIB-Att attribute shall be processed in order of descending length: that is a match with a longer prefix shall take precedence over a match with a shorter prefix.

9.4 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, and shall proceed 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 is located in another routeing domain, then the NPDU shall be forwarded directly to the BIS indicated in the NEXT_HOP entry. (The next-hop BIS is in a different routeing domain, and is on the same subnetwork as the local BIS.)

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 is located in the local BIS′s routeing domain, then the local BIS has the following options:

   1) **Forward on a Real Inter-domain Link:** The local BIS may forward the NPDU directly to the next-hop BIS if the next-hop BIS and the local BIS are attached to a common subnetwork.

      If the local BIS determines a neighbor IS (on a path to the next-hop BIS) is also a BIS, the local BIS may forward the NPDU directly to that system without encapsulation.

   2) **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. The QOS parameter of the encapsulating (outer) NPDU shall be set equal to the QOS parameter of the encapsulated (inner) NPDU. The encapsulated NPDU shall then be handed over to the intra-domain routeing protocol.

Table 2. 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
1	1	-	-	-	0	0	1	Capacity
1	1	-	-	-	1	1	0	Capacity
0	1	0	0	0	0	0	0	Source Specific QOS
1	0	0	0	0	0	0	0	Destination Specific QOS

Note: 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.

3) **Use Paths Calculated by the Intra-domain 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[9]. If there is an intra-domain route that supports the QOS Maintenance parameter of the NPDU, then the NPDU may be forwarded along this route.

Note: 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.

10. Interface to ISO 8473

The protocol protocol described in this international standard interfaces at its lower boundary to ISO 8473 using the ISO 8473 primitives shown in Table 3.

The parameters associated with the N-UNITDATA request primitive are:

— *Destination NET* is the NET of the BIS to which this BISPDU is being sent (that is, the remote BIS)

— *Source NET* 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 NET* is the NET of the BIS to which this BISPDU is being sent

— *Source NET* 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

[9] 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 3. IDRP-CL Primitives

Primitive	Parameters
N-UNITDATA Request	Destination NET Source NET QOS Userdata
N-UNITDATA Indication	Destination NET Source NET QOS Userdata

— *Userdata* is an ordered sequence of octets which comprise the BISPDU being delivered to the local BIS

11. Constants

This constants used by the protocol defined in this international standard are enumerated in Table 4.

12. 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 manage-ment specification for IDRP, expressed in the GDMO notation defined in ISO 10165-4.

12.1 Name Bindings

iSOxxxx-NB **NAME BINDING**

SUBORDINATE OBJECT CLASS idrp_config
NAMED BY
  SUPERIOR OBJECT CLASS ″ISO/IEC xxxx″: networkEntity;
  WITH ATTRIBUTE ″ISO/IEC xxxx″: idrp_config_MO_Name
  CREATE with-automatic-instance-naming iSO-xxxxx-NB-pl;
  DELETE on-if-no-contained-objects;
REGISTERED AS {ISO xxxxx-IDRP.nboi ISOxxxx-NB (1)};

adjacentBIS **NAME BINDING**

SUBORDINATE OBJECT CLASS adjacentBIS
NAMED BY
  SUPERIOR OBJECT CLASS idrp_config
  WITH ATTRIBUTE BIS-NET;
  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 {ISO xxxx-IDRP.nboi adjacentBIS (2)};

12.2 Local BIS Managed Objects for IDRP

idrp_config **MANAGED OBJECT CLASS**

DERIVED FROM ″ISO/IEC xxxxxx″: top
CHARACTERIZED BY localbispackage **PACKAGE**
ATTRIBUTES
  INTERNAL-BIS **GET**,
  INTRA-IS **GET**,
  EXTERNAL-BIS-Neighbors **GET**,
  INTERNAL-SYSTEMS **GET**,
  Local-RDI **GET**,
  RDC-Config **GET**,
  LocalSNPAs **GET**,
  MultiExit **GET**,
  Server **GET**,
  MaximumPDUsize **GET**,
  HoldTime **GET**,
  OustandingPDUs **GET**,
  AuthenticationCode **GET**,
  RetransmissionTimer **GET**,
  CloseWaitDelayPeriod **GET**,
  RDTransitDelay **GET**,
  RDLRE **GET**,
  LocExpense **GET**,
  RIBAttsSet **GET**,
  Capacity **GET**,
  Priority **GET**;
  version **GET**
  maxRIBIntegrityCheck **GET**

ACTIONS
  startevent,
  stopevent;
NOTIFICATIONS
  enterFSMState,
  FSMStateChange,
  errorBISPDUsent,
  openBISpduRDCerror,
  errorBISPDUconnectionclose;
  CorruptAdjRIBIn
  packetbomb
REGISTERED AS {ISOxxxx-IDRP.moi idrp_config (1) ;;;

Table 4. Architectural Constants of IDRP

Name of Constant	Value	Description
Inter-domain Routeing Protocol Identifier	--	The SPI for the protocol described in this international standard
MinBISPDULength	31	The size in octets of the smallest allowable BISPDU.
MinRouteSelectionInterval	30 minutes	The minimum time between successive UPDATE PDUs advertising feasible routes learned from other RDs
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 8.16.3.3.
MaxCPUOverloadPeriod	1 hour	Maximum time in which a BIS can remain CPU-overloaded before terminating its BIS-BIS connections.

12.3 Adjacent BIS Peer Managed objects

adjacentBIS **MANAGED OBJECT CLASS**

```
    DERIVED FROM ″ISO/IEC xxxxx″: top
    CHARACTERIZED BY adjacentBIS PACKAGE
    ATTRIBUTES
        BIS_NET GET,
        BIS_RDI GET,
        BIS RDC GET,
        BISnegotiatedversion GET,
        ISpeerSNPAs GET,
        Authentication_type GET,
        State GET,
        Lastseqnosent GET,
        Lastseqnorecv GET,
        Lastacksent GET,
        Lastackrecv GET,
        updatesIn  GET,
        updatesOut GET,
        totalBISPDUsIn GET,
        totalBISPDUsOut GET,
        KeepalivesSinceLastUpdate GET;
    ATTRIBUTE GROUPS
        counters
            updateIN
            updateOUT
            totalBISPDUsIN
            totalBISPDUsOUT
            KeepalivesSinceLastUpdate;
        stateinfo
            state
            lastseqnosent
            lastseqnorecv
            lastacksent
            lastackrecv;
    REGISTERED AS [ISO xxxxx-IDRP.moi
    adjacentBIS(2);
```

12.4 Attribute Definitions

InternalBIS **ATTRIBUTE**

```
    WITH ATTRIBUTE SYNTAX
    ISOXXXX-IDRP.BIS_group;
    MATCHES FOR Equality;
    BEHAVIOUR InternalBIS-B
        BEHAVIOUR DEFINED AS The set of NETs
        which identify the BISs in this routeing
        domain;
    REGISTERED AS {ISOXXXX-IDRP.aoi
    InternalBIS(1);
```

IntraIS **ATTRIBUTE**

```
    WITH ATTRIBUTE SYNTAX
    ISOXXXX-IDRP.BIS_group;
    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 {ISOXXXX-IDRP.aoi IntraBIS(2);
```

ExternalBISNeighbor **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOXXXX-IDRP.BIS_group;
MATCHES FOR Equality;
BEHAVIOUR ExternalBISNeighborB
BEHAVIOUR DEFINED AS The set of NETs which identify the BISs in adjacent routeing domain that are reachable via a single sub-network hop.
REGISTERED AS {ISOXXXX-IDRP.aoi ExternalBISNeighbor (3);

InternalSystems **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOXXXX-IDRP.system_id_group
MATCHES FOR Equality;
BEHAVIOUR InternalSystems-B
BEHAVIOUR DEFINED AS The set of NETs and NSAPS which identify the systems in this routeing domain which the BIS uses to construct network layer reachability information;
REGISTERED AS ISOXXXX-IDRP.aoi InternalSystems (4);

LocalRDI **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOXXXX-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 ISOXXXX-IDRP.aoi LocalRDI (5);

RDC-Config **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOXXXX-IDRP.rdc_group
MATCHES FOR Equality;
BEHAVIOUR RDC-Config-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 ISOXXXX-IDRP.aoi RDC-Config (6);

LocalSNPA **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOXXXX-IDRP.localSNPA
MATCHES FOR Equality;
BEHAVIOUR localSNPA-B
BEHAVIOUR DEFINED AS The list of SNPAs of this BIS;
REGISTERED AS ISOXXXX-IDRP.aoi LocalSNPA(7);

Multiexit **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX 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 ISOXXXX-IDRP.aoi MultiExit(8);

routeserver **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX 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 ISOXXXX-IDRP.aoi routeserver(8);

maximumPDUsize **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.MaximumPDUSize;
MATCHES FOR Equality, Ordering;
BEHAVIOUR maximumPDUsize-B
BEHAVIOUR DEFINED AS The maximum number of octets that this BIS is able to handle in an incoming BISPDU;
REGISTERED AS ISOXXXXX-IDRP.aoi maximumPDUsize(9);

holdtime **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-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 ISOXXXX-IDRP.aoi holdtime(10);

outstandingPdus **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-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 ISOXXXX-IDRP.aoi outstandingPdus(11);

authenticationCode **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.AuthenicationCode;
MATCHES FOR Equality, Ordering;
BEHAVIOUR authenticationCode-B
BEHAVIOUR DEFINED AS Indication of which authentication mechanism will be used;
REGISTERED AS ISOXXXX-IDRP.aoi authenticationCode (12);

RetransmissionTime **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.retransmissiontimer
MATCHES FOR Equality, Ordering;
BEHAVIOUR RetransmissionTimer-B
BEHAVIOUR DEFINED AS The Number of seconds of between KEEPALIVE messages if no other traffic is sent;
REGISTERED AS ISOXXXX-IDRP.aoi RetransmisionTimer (13);

CloseWaitDelayPeriod **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.closewaitdelayperiod
MATCHES FOR Equality, Ordering;
BEHAVIOUR CloseWaitDelayPeriod-B
BEHAVIOUR DEFINED AS The number of seconds the local system shall stay in the CLOSE-WAIT state prior to changing to the CLOSED stated.;
REGISTERED AS ISOXXXX-IDRP.aoi CloseWaitDelayPeriod (14);

RDTransitDelay **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.RDtransitdelay
MATCHES FOR Equality, Ordering;
BEHAVIOUR RDTRansitDelay-B
BEHAVIOUR DEFINED AS The estimated average delay across a Routeing Domain in units of 500ms.
REGISTERED AS ISOXXXX-IDRP.aoi RDTransitDelay (15);

RDLRE **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.rdlre
MATCHES FOR Equality, Ordering;
BEHAVIOUR RDLRE-B
BEHAVIOUR DEFINED AS The average error rate of a Routeing Domain in units of an integer which if divided by 2**32-1 will provided the actual probability of the error.
REGISTERED AS ISOXXXX-IDRP.aoi RDLRE(16);

LocExpense **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.locexpense
MATCHES FOR Equality, Ordering;
BEHAVIOUR LocExpense-B
BEHAVIOUR DEFINED AS The monetary expense of transiting this Routeing Domain. The attribute contains an indication of cost and the units in which it is calculated;
REGISTERED AS ISOXXXX-IDRP.aoi LocExpense(17);

RIBAttsSet **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.ribattsSet
MATCHES FOR Equality;
BEHAVIOUR RIBAttsSet-B
BEHAVIOUR DEFINED AS The set of Rib Attributes supported by this BIS.;
REGISTERED AS ISOXXXX-IDRP.aoi RIBAttsSet(18);

Capacity **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.capacity
MATCHES FOR Equality, Ordering;
BEHAVIOUR Capacity-B
BEHAVIOUR DEFINED AS The traffic carrying capacity of this Routeing Domain.
REGISTERED AS ISOXXXX-IDRP.aoi Capacity(19);

Priority **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-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 ISOXXXX-IDRP.aoi Priority(20);

BIS_NET **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.bis_net;
MATCHES FOR Equality;
BEHAVIOUR BIS_NET-B
BEHAVIOUR DEFINED AS The NET of the remote BIS of this BIS to BIS connection.;
REGISTERED AS {ISO-IDRP.aoi BIS_NET (21)};

BIS_RDI **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.rdi;
MATCHES FOR Equality;
BEHAVIOUR BIS_RDI-B
BEHAVIOUR DEFINED AS The RDI of the remote BIS of this BIS to BIS connection.;
REGISTERED AS {ISO-IDRP.aoi BIS_RDI (22)};

BIS_RDC **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.rdc_group
MATCHES FOR Equality;
BEHAVIOUR BIS_RDC-B

BEHAVIOUR DEFINED AS The RDC the remote BIS belongs to in this BIS to BIS connection.;

REGISTERED AS {ISO-IDRP.aoi BIS_RDC (23)};

BISnegotiatedversion **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.bisnegotiatedvesion;
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 {ISOxxxx-IDRP.aoi BISnegotiatedversion (24)};

BISpeerSNPAs **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-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 {ISOxxxx-IDRP.aoi BISpeerSNPAs (25)};

Authentication_type **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.auth_type
MATCHES FOR Equality, Ordering;
BEHAVIOUR authentication_type-B
BEHAVIOUR DEFINED AS The authentication type the remote BIS sent in the OPEN BISPDU in this BIS to BIS connection.
REGISTERED AS {ISOxxxx-IDRP.aoi Authentication_type (26)};

State **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-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 {ISOxxxx-IDRP.aoi state (27)};

Lastseqnosent **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.lastseqnosent
DERIVED FROM nonWrappingCounter;
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 {ISOxxxx-IDRP.aoi Lastseqnosent (28)};

Lastseqnorecv **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISOxxxx-IDRP.lastseqnorecv
DERIVED FROM nonWrappingCounter;
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 {ISO xxxx-IDRP.aoi Lastseqnorecv (28)};

Lastacksent **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.lastacksent
DERIVED FROM nonWrappingCounter;
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 {ISO xxxxx-IDRP.aoi Lastacksent (29)};

Lastackrecv **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.lastackrecv
DERIVED FROM nonWrappingCounter;
MATCHES FOR Equality, Ordering;
BEHAVIOUR Lastacksent-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 {ISO xxxxx-IDRP.aoi Lastackrecv (30)};

updatesIn **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.updatesin
DERIVED FROM nonWrappingCounter;
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 {ISO xxxx-IDRP.aoi updatesIn (31)};

updatesOut **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.updatesout
DERIVED FROM nonWrappingCounter;
MATCHES FOR Equality, Ordering;
BEHAVIOUR updatesOut-B
BEHAVIOUR DEFINED AS The number of UPDATE BISPDUs sent by this BIS on this BIS to BIS connection.
REGISTERED AS {ISO xxxx-IDRP.aoi updatesOut (32)};

totalBISPDUsIn **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.totalbispdusin
DERIVED FROM nonWrappingCounter;
MATCHES FOR Equality, Ordering;
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 {ISO xxxx-IDRP.aoi totalBISPDUsIn (33)};

totalBISPDUsOut **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.totalbispdusout
DERIVED FROM nonWrappingCounter;
MATCHES FOR Equality, Ordering;
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 {ISO xxxx-IDRP.aoi totalBISPDUsOut (34)};

KeepalivesSinceLastUpdate **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxxx-IDRP.keepaliveSincelastupdate
DERIVED FROM nonWrappingCounter;
MATCHES FOR Equality, Ordering;
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 {ISO xxxx-IDRP.aoi KeepAlivesSinceLastUpdate (35)};

version **ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.version
MATCHES FOR Equality, Ordering;
BEHAVIOUR version-B
BEHAVIOUR DEFINED AS The version of IDRP protocol this machine defaults to using.;
REGISTERED AS {ISO xxxx-IDRP.aoi version (36)};

maxRIBIntegrityCheck**ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-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 {ISO xxxx-IDRP.aoi MaxRIBIntegrityCheck(37)};

maxRIBIntegrityTimer**ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.ribintegritytimer
DERIVED FROM timer
MATCHES FOR Equality, Ordering;
BEHAVIOUR RIBIntegritytimer-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 {ISO xxxx-IDRP.aoi MaxRIBIntegrityTimer(37)};

closeWaitDelayTimer**ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.waitdelaytimer
DERIVED FROM timer
MATCHES FOR Equality, Ordering;
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 {ISO xxxx-IDRP.aoi CloseWaitDelayTimer(38)};

keepAliveTimer**ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.keepalivetimer
DERIVED FROM timer
MATCHES FOR Equality, Ordering;
BEHAVIOUR Keepalivetimer-B
BEHAVIOUR DEFINED AS The timer that measures in seconds the time that has elapsed since the previous KEEPALIVE PDU was received by the local BIS. Upon its expiration, the BIS will send a BISPDU to its peer BIS;
REGISTERED AS {ISO xxxx-IDRP.aoi KeepAliveTimer(39)};

minRouteSelectionTimer**ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.routeselectiontimer
DERIVED FROM timer
MATCHES FOR Equality, Ordering;
BEHAVIOUR Routeselectiontimer-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 8.16.3.1;
REGISTERED AS {ISO xxxx-IDRP.aoi MinRouteSelectiontimer(40)};

minRDOriginationTimer**ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.rdoriginationtimer
DERIVED FROM timer
MATCHES FOR Equality, Ordering;
BEHAVIOUR RDOriginationtimer-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 8.16.3.2;
REGISTERED AS {ISO xxxx-IDRP.aoi MinRDOriginationtimer(41)};

maxCPUOverloadTimer**ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.maxcpuoverloadtimer
DERIVED FROM timer
MATCHES FOR Equality, Ordering;
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 Appendix F;
REGISTERED AS {ISO xxxx-IDRP.aoi MaxCPUOverloadtimer(42)};

12.5 Action Definitions

minRDOriginationTimer**ATTRIBUTE**

WITH ATTRIBUTE SYNTAX ISO xxxx-IDRP.rdoriginationtimer
MATCHES FOR Equality, Ordering;
BEHAVIOUR RDOriginationtimer-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 8.16.3.2;
REGISTERED AS {ISO xxxx-IDRP.aoi MinRDOriginationtimer(40)};

startevent **Action**

BEHAVIOUR
startevent **BEHAVIOUR**
MODE CONFIRMED;
CONTEXT ACTION-INFO;
WITH INFORMATION SYNTAX ISO xxxx-idrp.Actioninfo;
WITH REPLY SYNTAX ISO xxxx-idrp.Startevenreply;
DEFINED AS The request to start communication with a remote BIS peer;
PARAMETERS Remotebis-NET;
MODE CONFIRMED;
REGISTERED AS ISO xxxxx-IDRP.aci startevent (1);

Stopevent **Action**

BEHAVIOUR
stopevent **BEHAVIOUR**
MODE CONFIRMED;
CONTEXT ACTION-INFO;
WITH INFORMATION SYNTAX ISO xxxx-idrp.Actioninfo;
WITH REPLY SYNTAX ISO xxxx-idrp.Stopevenreply;
PARAMETERS Remotebis-NET;
MODE CONFIRMED;
DEFINED AS The request to stop communication with a remote BIS peer;
REGISTERED AS ISO xxxxx-IDRP.aci stopevent (2);

12.6 Notification Definitions

enterFSMstatemachine **NOTIFICATON**

BEHAVIOUR enterFSMstatemachine-B
BEHAVIOUR DEFINED AS The indication of starting the FSM state machine to establish a connection with a remote BIS.;
MODE NON-CONFIRMED
PARAMETERS Remotebis-NET;
WITH INFORMATION SYNTAX ISOxxxx-IDRP.NotificationInfo
REGISTERED AS ISOxxxx-IDRP.noi enterFSMstatemachine (1);

FSMstatechange **NOTIFICATION**

BEHAVIOUR FSMstatechange-B
BEHAVIOUR DEFINED AS The indication of transiting from one state to another in the IDRP connection state machine in communication with another BIS.;
MODE NON-CONFIRMED
PARAMETERS remoteBIS-NET, state;
WITH INFORMATION SYNTAX ISOxxxx-IDRP.NotificationInfo
REGISTERED AS ISOxxxx-IDRP.noi FSMstatechange(2);

errorBISPDUsent **NOTFICATION**

BEHAVIOUR errorBISPDUsent-B
BEHAVIOUR DEFINED AS The indication of an error in the format of BISPDU.;
MODE NON-CONFIRMED
PARAMETERS Remotebis-NET, BISpduerrorcode, BISerrorsubcode, BISpduerrorinfo;
WITH INFORMATION SYNTAX ISOxxxx-IDRP.NotificationInfo **REGISTERED AS** ISOxxxx-IDRP.noi errorBISPDUsent (3);;

openBISpduRDCerror **NOTIFICATION**

BEHAVIOUR openBISpduRDCerror-B
BEHAVIOUR DEFINED AS The indication that an OPEN PDU has been received with the RDC Config for remote BIS and this BIS do

not indicate that the two BIS trying to establish a connection are a part of the same confederations;
MODE NON-CONFIRMED
PARAMETERS Remotebis-NET, RemoteRDCconfig, LocalRDCConfig;
WITH INFORMATION SYNTAX ISOxxxx-IDRP.NotificationInfo
REGISTERED AS ISOxxxx-IDRP.noi errorpduRDCerror(4);

errorBISPDUconnectionclose **NOTIFICATION**

BEHAVIOUR errorBISPDUconnectionclose-B
BEHAVIOUR DEFINED AS The indication that an ERROR BISPDU has been received from a remote BIS;
MODE NON-CONFIRMED
PARAMETERS Remotebis-NET, bispduerrorcode, bispduerrorsubcode,bispduinfo;
WITH INFORMATION SYNTAX ISOxxxx-IDRP.NotificationInfo
REGISTERED AS ISOxxxx-IDRP.noi errorBISPDUconnectionclose(5);;

CorruptAdjRIBIn **NOTIFICATION**

BEHAVIOUR corruptAdjRIBIn-B
BEHAVIOUR DEFINED AS The indication that the local method of checking the Adj-RIB-In has found an error. All Adj-RIBs-In are being purged.
MODE NON-CONFIRMED
PARAMETERS maxAdjRibIntegritycheck;
WITH INFORMATION SYNTAX ISOxxxx-IDRP.NotificationInfo
REGISTERED AS ISOxxxx-IDRP.noi corruptAdjRIBIn(6);;

packetBomb **NOTIFICATION**

BEHAVIOUR packetBomb-B
BEHAVIOUR DEFINED AS The indication that the local BIS has been presented with a BISPDU whose source is not one of the BISs adjacent to the local BIS. Such BISPDUs are rejected by the local BIS.
MODE NON-CONFIRMED
WITH INFORMATION SYNTAX ISOxxxx-IDRP.NotificationInfo
REGISTERED AS ISOxxxx-IDRP.noi packetBomb(7);;

12.7 Parameter Definitions

RemoteBIS-NET **PARAMETER**

CONTEXT ACTION-REPLY;
WITH SYNTAX ISOxxxx-IDRP.remoteBIS-NET;
BEHAVIOUR RemoteBIS-NET-B
PARAMETER DEFINED AS The NET of the Remote BIS that this local BIS is starting IDRP protocol communication with.;
REGISTERED AS ISOxxx-IDRP.proi RemoteBIS-NET(1);

Remotebis-NET **PARAMETER**

CONTEXT EVENT-INFO;
WITH SYNTAX ISOxxxx-IDRP.remoteBIS-NET;
BEHAVIOUR Remotebis-NET-B
PARAMETER DEFINED AS The NET of the Remote BIS that this local BIS is starting IDRP protocol communication with.;
REGISTERED AS ISOxxxx-IDRP.proi Remotebis-NET(1);

STATE **PARAMETER**

CONTEXT EVENT-INFO;
WITH SYNTAX ISOxxxx-IDRP.state
BEHAVIOUR ISOxxx-IDRP.STATE-B
PARAMETER DEFINED AS The state of the local BIS Finite State machine.;
REGISTERED AS ISOxxxx-IDRP.prio STATE(1);

BISpduerrorcode **PARAMETER**

CONTEXT EVENT-INFO;
WITH SYNTAX ISOxxxx-IDRP.bispduerrorcode
BEHAVIOUR ISOxxxx-IDRP.BISpduerrorcode-B
BEHAVIOUR DEFINED AS The error code indicating what type of error occurred in the BIS PDU.;
REGISTERED AS ISOxxxx-IDRP.prio BISpduerrorcode(2)

BISpduerrorsubcode **PARAMETER**

CONTEXT EVENT-INFO;
WITH SYNTAX ISOxxxx-IDRP.bispduerrorsubcode
BEHAVIOUR ISOxxxx-IDRP.BISpduerrorcode-B
BEHAVIOUR DEFINED AS The error code indicating what type of error within the major error type occurred in the BIS PDU.;
REGISTERED AS ISOxxxx-IDRP.prio BISpduerrorsubcode(3)

BISpduerrorinfo **PARAMETER**

CONTEXT EVENT-INFO;
WITH SYNTAX ISOxxxx-IDRP.bispduerrorinfo
BEHAVIOUR ISOxxxx-IDRP.BISpduerrorinfo-B
BEHAVIOUR DEFINED AS The additional information from original pdu that indicated an error in the BIS PDU.;
REGISTERED AS ISOxxxx-IDRP.prio BISpduerrorinfo(4);

RemoteRDCconfig **PARAMETER**

CONTEXT EVENT-INFO;
WITH SYNTAX ISOxxxx-IDRP.remoteRDCconfig;
BEHAVIOUR ISOxxxx-IDRP.RemoteRDCconfig-B
BEHAVIOUR DEFINED AS The Routing Domain Confederation (RDC) information from the remote BIS on this BIS to BIS communication.;

REGISTERED AS ISOxxxx-IDRP.prio RemoteRDCconfig(5);

LocalRDCconfig **PARAMETER**

CONTEXT EVENT-INFO;
WITH SYNTAX ISOxxxx-IDRP.localRDCconfig;
BEHAVIOUR ISOxxx-IDRP.LocalRDCconfig-B
BEHAVIOUR DEFINED AS The Routing Domain Confederation (RDC) information from this local BIS on this BIS to BIS communcation.;
REGISTERED AS ISOxxxx-IDRP.prio LocalRDCconfig(6);

12.8 Attribute Groups

counters **ATTRIBUTE** group

DESCRIPTION The group of all counter per BIS connection
REGISTERED AS {ISO xxxxx-IDRP.agoi counters [1]};

stateinfo **ATTRIBUTE** group

DESCRIPTION The group of all state information per BIS connection
REGISTERED AS {ISO xxxx-IDRP.agoi stateinfo[2]};

12.9 ASN.1 MODULES

```
ISO XXXXX-IDRP(tbd1) DEFINITIONS::=BEGIN
  -- object identifier definitions
sc6 OBJECT IDENTIFIER ::= {joint-iso-ccitt sc6(?)}
  -- value to be assigned by SC21 secretariat
idrpoi OBJECT IDENTIFIER ::= {sc6 iso xxxxx(?)}
  -- value to be assigned by SC6 secretariat
moi OBJECT IDENTIFIER ::= {idrpoi objectClass (3)}
poi OBJECT IDENTIFIER ::= {idrpoi package (4)}
proi OBJECT IDENTIFIER ::={idrpoi parameter(5)}
nboi OBJECT IDENTIFIER ::={idrpoi nameBinding (6)}
aoi OBJECT IDENTIFIER ::={idrpoi attribute (7)}
agoi OBJECT IDENTIFIER ::={idrpoi attributeGroup (8)}
aoi OBJECT IDENTIFIER ::={idrpoi action (9)}
noi OBJECT IDENTIFIER ::={idrpoi action (10)}
ActionInfo ::=SET OF Parameter
ActionReply ::= SEQUENCE {
  responseCode OBJECT IDENTIFIER
  responseArgs SET of Parameter OPTIONAL}
AuthenticationCode ::=ENUMERATED{
  integrityOnly(0),
  integrityPlusAuthentication(1)}
auth_type ::=AuthenticationCode
BIS_group ::= SET OF {NetworkEntityTitle}
bis_net::= NetworkEntityTitle
bisnegotiatedversion ::=version
bispduerrorcode::= ENUMERATED {
  OPEN_PDU_Error (1),
  UPDATE_PDU_Error (2),
  Hold_timer_Expired (3),
bispduerrorsubcode ::= SET OF {
  openerrorsubcode,
  updateerrorsubcode}
bispduerrorinfo ::=OCTETSTRING(1...50)
bispeersSNPAs ::= SNPAAddresses
Boolean ::= BOOLEAN
capacity ::==INTEGER(1...255)
closewaitdelayperiod ::=INTEGER(150)
Holdtime ::=INTEGER(1...65 535)
keepaliveSincelastupdupdate ::=INTEGER(1...65 535)
keepalivetimer ::= timer
lastseqnosent ::=INTEGER(1...(4 294 967 295))
lastseqnorecv ::=INTEGER(1...(4 294 967 295))
lastacksent ::=INTEGER(1...(4 294 967 295))
lastackrecv ::=INTEGER(1...(4 294 967 295))
locexpense ::= INTEGER(1...65 535)
localRDCconfig ::=rdc_group
local_SNPAs ::= SNPAaddresses
MaximumPDUSize ::=INTEGER(1..65 535)
NetworkEntityTitle ::=OCTETSTRING(SIZE(1...20))
NotificationInfo ::=SET OF Parameter
openerrorsubcode ::=ENUMERATED {
  UnsupportedVersion_number (1),
  Bad_Max_PDU_size (2),
  Bad_Outstanding_PDUs (3),
  Bad_Peer_RD (4),
  Unsupported_Authentication_code (5),
  Authentication_Failure (6),
  Bad_RIB-AttrsSet (7),
  RDC_mismatch (8)}
OutstandingPdus ::=INTEGER(0...255)
Parameter ::= SEQUENCE {
  paramID OBJECTIDENTIFIER
  paramInfo ANY DEFINED BY ParmID}
Priority ::=INTEGER(0...14)
rdi ::=OCTETSTRING(SIZE(1...20));
  --assigned from the NSAP address space
rdc_group::=SEQUENCE{SEQUENCE
  rdc_set_id,SET OF {rdi}}}
rdc_set_id ::=INTEGER(1..255)
RDtransitDelay ::=INTEGER(0...65 535)
rdlre ::=INTEGER(0...(4 294 967 295))
retransmission_timer ::= timer
remoteBIS-NET ::=NetworkEntityTitle
remoteRDCconfig ::=rdc_group
ribattsSet ::=SEQUENCE {
  SEQUENCE  {
   ribsetid,
   ribsetcount,
   SET OF {rib_attributes}}
ribsetid ::=INTEGER(1..255)
ribsetcount ::=INTEGER(0..255)
rib_attributes ::= SEQUENCE OF {
  rib_attribute,
  rib_value}
rib_attribute ::= ENUMERATED {
  TRANSIT_DELAY (1),
  RESIDUAL_ERROR (2),
  EXPENSE (3),
  SourceSpecificQOS (4),
  DestinationSpecificQOS (5),
  SourceSpecificSecurity (6),
  DestinationSpecificSecurity(7),
  Capacity (8),
  Priority (9)}
rib_value :: = OCTETSTRING
 -- This octetstring may vary according to the
 -- rib_attribute value.  Source Specific QOS,
 -- Destination SPecific QOS, SOurce Specific
 -- Security, Destination Specific Security,
 -- may have varying lengths of rib attribute
values.
 -- See the appropriate subclause of 8.12
 -- for more details
routeselectiontimer ::= timer
rdoriginationtimer ::= timer
SNPAAddress ::=SET OF {
  SNPA_Type, SNPAaddress}
SNPAaddress ::= SEMIOCTET 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)}
system_id_group ::= SEQUENCE OF {
  SET OF {NetworkEntityTitle},
  SET OF {EndSystemNSAPs}}
timer ::=SEQUENCE {
  exponent [1] INTEGER (-62...63)
  mantissa [2] INTEGER (0...65 535)
updateerrorsubcode ::=ENUMERATED {
  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)}
updatesin ::=INTEGER(1...4 294 967 295)
updatesout ::=INTEGER(1...4 294 967 295)
totalbispdusin ::=INTEGER(1..4 294 967 295)
totalbispdusout ::=INTEGER(1..4 294 967 295)
version ::=INTEGER (1...255)
waitdelattimer ::= timer
```

13. 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 Appendix A.

Note: 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.

13.1 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 structure and format of 7

b) transmitting and receiving NSAP prefixes that have been encoded as single values according to 8.1.2.1

c) generating, recognizing upon receipt, and updating each of the following well-known mandatory path attributes as described in the indicated clauses:

— RD_PATH in 8.13.2

— RD_HOP_COUNT 8.13.3

d) recognizing upon receipt and correctly updating 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 8.13.1

— NEXT_HOP in 8.13.3

— UNREACHABLE in 8.13.4

— DIST_LIST_INCL in 8.13.5

— DIST_LIST_EXCL in 8.13.6

— LOCAL_PREF in 8.13.8

— TRANSIT DELAY in 8.13.9

— RESIDUAL ERROR in 8.13.10

— EXPENSE in 8.13.11

— SOURCE SPECIFIC QOS in 8.13.12

— DESTINATION SPECIFIC QOS in 8.13.13

— HIERARCHICAL RECORDING in 8.13.14

— SOURCE SPECIFIC SECURITY in 8.13.17

— DESTINATION SPECIFIC SECURITY in 8.13.18

— CAPCITY in 8.13.19

— PRIORITY in 8.13.20

e) utilizing domain configuration information in the format described in 8.3

f) providing reliable delivery of BISPDUs using the methods of 8.5

g) establishing, closing, and maintaining BIS-BIS connections in accordance with the procedures of 8.6

h) responding to error conditions in BISPDUs according to 8.19

i) negotiating the protocol version number according to 8.7

j) operating in a ″fail-stop″ manner in regard to corrupted routeing information, according to 8.10.2

k) maintaining RIBs as described in 8.10.

l) handling incoming BISPDUs as described in 8.15.

m) detecting inconsistent routeing information in accordance with 8.15.2.

n) receiving and recognizing information which has been reduced in size according to the methods of 8.17.4.

o) distributing network reachability information within a routeing domain according to 8.16.1.

p) distributing network reachability information outside a routeing domain according to 8.16.2.

q) selecting RD-paths according to 8.17.3.

r) detecting mutually unsatisfiable routeing policies according to 8.11.

s) forwarding ISO 8473 NPDUs according to 9.

t) supporting the interface to ISO 8473 using the service primitives according to 10.

u) providing the managed objects described in 12.

v) supporting the authentication mechanism described in 8.9.

13.2 Conformance to Optional Functions

13.2.1 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 8.17.4 and 8.17.5.

13.2.2 Supporting RDCs

A BIS that claims to support Routeing Domain Confederations shall construct them and distribute their routeing information in accordance with clause 8.14.

13.2.3 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:

EXT_INFO: 8.13.1

NEXT HOP: 8.13.3

UNREACHABLE: 8.13.4

DIST LIST INCL: 8.13.5

DIST LIST EXCL: 8.13.6

LOCAL PREF: 8.13.8

TRANSIT DELAY: 8.13.9

RESIDUAL ERROR: 8.13.10.

EXPENSE: 8.13.11.

SOURCE SPECIFIC QOS: 8.13.12.

DESTINATION SPECIFIC QOS: 8.13.13.

HIERARCHICAL RECORDING: 8.13.14

SOURCE SPECIFIC SECURITY: 8.13.17

DESTINATION SPECIFIC SECURITY: 8.13.18

CAPACITY: 8.13.19

PRIORITY: 8.13.20

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—or potential user—of 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: 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: 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 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.

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 10. Illustrative PICS for Conditional Items

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 10, 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 5. 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))	

Notes:

a) Only the first three items are required for all implementations; other information may be completed as appropriate in meeting the requirement for full identification.

b) 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 6. 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 7. PICS Proforma for IDRP: General

Item	Questions/Features	References	Status	Support
BASIC	Are all basic BIS functions implemented?	13.1	M	Yes __
BISMGT	Is this system capable of being managed by the specified management information?	12 .	M	Yes __
VERNEG	Does this BIS support version negotiation?	8.7	M	Yes__
HOPS	Does this BIS support the RD_HOP_COUNT attribute?	8.13.16	M	Yes__
PATH	Does this BIS support the RD_PATH attribute?	8.13.2.	M	Yes__
FSM	Does this BIS manage BIS-BIS connections according to the BIS FSM description?	8.6	M	Yes__
ERROR	Does this BIS handle error handling for IDRP?	8.19	M	Yes__
RIBCHK	Does this BIS operate in a "fail-stop" manner with respect to corrupted routeing information?	8.10.2	M	Yes__

A.4.4 PICS Proforma: IDRP Update Send Process

Table 8. PICS Proforma for IDRP: Update-Send

Item	Questions/Features	References	Status	Support
INT	Does this BIS provide the internal update procedures?	8.16.1	M	Yes __
RTSEL	Does this BIS support the **MinRouteSelectionInterval** timer?	8.16.3.1	M	Yes __
RTORG	Does this BIS support the **MinRDOriginationInterval** timer?	8.16.3.2	M	Yes __
JITTER	Does this BIS provide jitter on its timers?	8.16.3.3	M	Yes __

A.4.5 PICS Proforma: IDRP Update Receive Process

Table 9. PICS Proforma for IDRP: Update-Receive

Item	Questions/Features	References	Status	Support
INPDU	Does the BIS handle inbound BISPDUs correctly?	8.15	M	Yes __
INCONS	Does this BIS detect inconsistent routeing information?	8.15.2	M	Yes__

A.4.6 PICS Proforma: IDRP Decision Process

Table 10. PICS Proforma for IDRP: Decision

Item	Questions/Features	References	Status	Support
TIES	Does the BIS break ties between candidate routes correctly?	8.17.1	M	Yes __
RIBUPD	Does this BIS update the correct Loc-RIBs?	8.17.2	M	Yes__
AGGRT	Does this BIS support route aggregation?	8.17.5 to 8.17.5.3	O	Yes__ No __
LOCK	Does this BIS provide interlocks between its decision process and the updating of the information in its Adj-RIBs-In?	8.17.6	M	Yes__

A.4.7 PICS Proforma: IDRP Receive Process

Table 11. PICS Proforma for IDRP: Receive

Item	Questions/Features	References	Status	Support
RCV	Does the BIS process incoming BISPDUs and respond correctly to error conditions?	8.15, 8.19	M	Yes __

A.4.8 PICS Proforma: IDRP CLNS Forwarding

Table 12. 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?	9	M	Yes __
DATTS	Does the BIS correctly extract the NPDU-derived Distinguishing Attributes from an 8473 NPDU?	9.2	M	Yes __
MATCH	Does the BIS correctly match the NPDU-derived Distinguishing Attributes with the corresponding FIB-Atts?	9.3	M	Yes __
EXTFWD	Does the BIS correctly forward NPDUs with destinations outside its own routeing domain?	9.4	M	Yes __
INTFWD	Does the BIS correctly forward NPDUs with destinations inside its own routeing domain?	9.1	M	Yes __

A.4.9 PICS Proforma: IDRP Optional Transitive Attributes

Table 13. 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?	8.13.7	O	Yes __ No __
CO/CL	Does the BIS support the CO/CL PATH attribute?	8.13.15	O	Yes __ No __

A.4.10 PICS Proforma: IDRP Well Known Discretionary Attributes

Table 14. 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?	8.13.1	O	Yes __ No __
NHRS	Does the BIS support generation of the NEXT_HOP attribute in support of route servers?	8.13.3	O	Yes __ No __
NHSN	Does the BIS support generation of the NEXT_HOP attribute to advertise SNPAs?	8.13.3	O	Yes __ No __
DLI	Does the BIS support generation of the DIST_LIST_INCL attribute?	8.13.5	O	Yes __ No __
DLE	Does the BIS support generation of the DIST_LIST_EXCL attribute?	8.13.6	O	Yes __ No __
LPRF	Does the BIS support generation of the LOCAL_PREF attribute?	8.13.8	O	Yes __ No __
TDLY	Does the BIS support generation of the TRANSIT DELAY attribute?	8.13.9	O	Yes __ No __
RERR	Does the BIS support generation of the RESIDUAL ERROR attribute?	8.13.10	O	Yes __ No __
EXP	Does the BIS support generation of the EXPENSE attribute?	8.13.11	O	Yes __ No __
SQOS	Does the BIS support generation of the SOURCE SPECIFIC QOS attribute?	8.13.12	O	Yes __ No __
DQOS	Does the BIS support generation of the DESTINATION SPECIFIC QOS attribute?	8.13.13	O	Yes __ No __
HREC	Does the BIS support generation of the HIERARCHICAL RECORDING attribute?	8.13.14	O	Yes __ No __
SSEC	Does the BIS support generation of the SOURCE SPECIFIC SECURITY attribute?	8.13.17	O	Yes __ No __
DSEC	Does the BIS support generation of the DESTINATION SPECIFIC SECURITY attribute?	8.13.18	O	Yes __ No __
CAPY	Does the BIS support generation of the CAPACITY attribute?	8.13.19	O	Yes __ No __
PRTY	Does the BIS support generation of the PRIORITY attribute?	8.13.20	O	Yes __ No __

Table 15. PICS Proforma for IDRP: Receiving Well Known Discretionary Attributes

Item	Questions/Features	References	Status	Support
EXTR	Does the BIS support receipt and subsequent propagation of the EXT_INFO attribute?	8.13.1	M	Yes __ No __ X:
NHRSR	Does the BIS support receipt and subsequent propagation of the NEXT_HOP attribute in support of route servers?	8.13.3	M	Yes __ No __ X:
NHSNR	Does the BIS support receipt and subsequent propagation of the NEXT_HOP attribute to advertise SNPAs?	8.13.3	M	Yes __ No __ X:
DLIR	Does the BIS support receipt and subsequent propagation of the DIST_LIST_INCL attribute?	8.13.5	M	Yes __ No __ X:
DLER	Does the BIS support receipt and subsequent propagation of the DIST_LIST_EXCL attribute?	8.13.6	M	Yes __ No __ X:
LPRFR	Does the BIS support receipt and subsequent propagation of the LOCAL_PREF attribute?	8.13.8	M	Yes __ No __ X:
TDLYR	Does the BIS support receipt and subsequent propagation of the TRANSIT DELAY attribute?	8.13.9	M	Yes __ No __ X:
RERRR	Does the BIS support receipt and subsequent propagation of the RESIDUAL ERROR attribute?	8.13.10	M	Yes __ No __ X:
EXPR	Does the BIS support receipt and subsequent propagation of the EXPENSE attribute?	8.13.11	M	Yes __ No __ X:
SQOSR	Does the BIS support receipt and subsequent propagation of the SOURCE SPECIFIC QOS attribute?	8.13.12	M	Yes __ No __ X:
DQOSR	Does the BIS support receipt and subsequent propagation of the DESTINATION SPECIFIC QOS attribute?	8.13.13	M	Yes __ No __ X:
HRECR	Does the BIS support receipt and subsequent propagation of the HIERARCHICAL RECORDING attribute?	8.13.14	M	Yes __ No __ X:
SSECR	Does the BIS support receipt and subsequent propagation of the SOURCE SPECIFIC SECURITY attribute?	8.13.17	M	Yes __ No __ X:
DSECR	Does the BIS support receipt and subsequent propagation of the DESTINATION SPECIFIC SECURITY attribute?	8.13.18	M	Yes __ No __ X:
CAPYR	Does the BIS support receipt and subsequent propagation of the CAPACITY attribute?	8.13.19	M	Yes __ No __ X:
PRTYR	Does the BIS support receipt and subsequent propagation of the PRIORITY attribute?	8.13.20	M	Yes __ No __ X:

Annex B. 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.

B.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

B.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 C. 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 8.16.3.3. is shown in Figure 11.

Annex D. 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 Appendix B to the octet stream which is composed from the concatenation of the sorted feasible routes.

Annex E. 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.

```
CONSTANT
  Jitter=0.25 (as 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

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. There are several courses of action that could be taken:

a) Remove routes that are not currently in any of the 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 F. 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 received with the UNREACHABLE path attribute.

If a route received in the UPDATE_PDU has the UNREACHABLE 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 G. Syntax and Semantics for Policy

(Informative)

A major task in using IDRP is to assign a degree of preference to each available path. This degree of preference will generally be a function of the number of RDs in the path, properties of the specific RDs in the path, the origin of the route, and properties of the specific border router to be used in the first hop. This annex presents an example of how a routeing domain administrator might articulate this function.

In addition to controlling the selection of the best path to a given network, the network administrator must control the advertisement of this best path to neighboring RDs. Therefore, path selection and path distribution emerge as the two key aspects of policy expression in IDRP usage.

Since different aspects of one RD′s policy interact, and since the policies of different RDs interact, it is important to facilitate the analysis of such interactions by means of high-quality and consistent tools. There is also a need for tools to translate the expression of the network administrator′s policy to some technical mechanism within a BIS to implement that policy. These factors suggest that there should be a globally consistent way of describing policies. The syntax and semantics of these policies should be capable of expressing the path selection phase within the local RD as well as the path redistribution phase to other RDs.

Because it may be desirable to coordinate routeing policy at an external level, it may prove worthwhile to create a language to describe this information in a globally consistent way. Policies expressed in such a language could conceivably be used by some high-level tools to analyze the interaction among the routeing policies of different Routeing Domains. The following defines one possible syntax and semantics for describing RD path policies from the point of view of the local RD. Alternative syntaxes with equivalent richness of functionality are not precluded. Other mechanisms may be needed to provide a fully functional configuration language.

A complete RD path, supplied by IDRP, provides the most important mechanism for policy enforcement. Assigning a degree of preference to a particular RD path can be modelled as a matching between this path and one or more predefined RD path patterns. Each predefined RD path pattern has a degree of preference that will be assigned to any RD path that matches it. Since patterns are naturally expressed by regular expressions, one can use regular expressions over the alphabet of RDIs to define RD path patterns and, therefore, to formulate policies.

Since certain constructs occur frequently in regular expressions, the following notational shorthand (operators) is defined:

. matches any RDI. To improve readability, ″.″ can be replaced by ″any″ so long as this does not introduce ambiguity.

* a regular expression followed by * means zero or more repetitions

+ a regular expression followed by + means one or more repetitions

? a regular expression followed by ? means zero or one repetition

| alternation

() parentheses group subexpressions--an operator, such as * or +, works on a single element or on a regular expression enclosed in parentheses

{m,n} a regular expression followed by {m,n} (where m and n are both non-negative integers and m <= n) means at least m and at most n repetitions.

{m} a regular expression followed by {m} (where m is a positive integer) means exactly m repetitions.

{m,} a regular expression followed by {m,} (where m is a positive integer) means m or more repetitions.

Any regular expression is generated by these rules.

The Policy Based Routeing Language can then be defined as follows: :

a) <Policy-Based-Routeing> ::= { <policy-statement> }

   Semantics: each policy statement might cause a given possible IDRP advertisement (possibility) to be installed into the routeing table as the route to a given (set of) networks. Thus, an empty Policy-Based-Routeing means that no possibilities will be accepted.

b) <policy-statement> ::=
   <policy-expression>′=′<locpref-expression>,<multi_exit_disc>′;′

   Semantics: if a given possibility matches the policy-expression, then that possibility will be accepted with a degree of preference denoted by the integer value locpref-expression.

c) <policy-expression> ::=
   <policy-term> |
   <policy-term> <policy-operator> <policy-term>

d) <policy-term> ::=
   <destination-list> <RD-path> <origin> <distribution-list> |
   ′(′ <policy-expression> ′)′ |
   NOT <policy-expression> | <>

e) <policy-operator> ::= OR | AND

   Semantics: the intersection of the Network Layer Reachability information of a possibility and the destination-list must be non-empty; the RD-path of the possibility must match the RD-path as a sequence; the origin of the possibility must be a member of the origin set; if these conditions are met, the route denoted by the possibility is accepted as a possible route to those NSAP address prefixes of the intersection of the possibility Network Layer Reachability information and the destination-list.

f) <RD-path> ::=
   ″regular expression over RD-path-segments″

   Semantics: the RD-path of the possibility must be generated by the regular expression <RD-path>.

g) <destination-list> ::=
   ′<′ {NSAP-address-prefix destination-list }′>′ |
   ′<′ ANY ′>′

   Semantics: A non-empty sequence enumerates the NSAP address prefixes of the destination-list; ANY denotes the set of all NSAPs.

h) <origin> ::= IDRP | EXTERNAL | ANY

   Semantics: origin enumerates the sources from which routeing information can be learned by the protocol; ANY denotes the set of all origins.

i) <distribution-list> ::=
   ′<′ { RDI } ′>′ |′<′ ANY ′>′

   Semantics: if a given possibility as accepted and installed into the routeing table, then distribution-list is the set of (neighboring) RDIs to whose border routers we will distribute the IDRP-derived routes.

j) <locpref-expression> ::= <locpref-term> |
   <locpref-term> ′+′ <locpref-term> |
   <locpref-term> ′-′ <locpref-term> |
   <locpref-term> ′*′ <locpref-term> |
   <locpref-term> ′/′ <locpref-term> |
   REJECT

   <locpref-term> ::= <integer> |
   <function> |
   ′(′ <locpref-expression> ′)′

   Semantics: if a possibility matches with the local preference REJECT, then that possibility will not be used. Otherwise, the integer value of the local preference indicates the degree of preference of the possibility, with higher values preferred over lower ones.

k) <Multi_exit_disc_pref> ::= ″True|False″

   Semantics: If the Multi_exit_disc_pref is set to true, and if two routes have the same path attriubtes except for MULTI_EXIT_DISC, and if the two policy expressions are evaluated to have the same preferences, then the MULTI_EXIT_DISC value received from the adjacent RD for each route is used to select between the two otherwise equally preferable routes. Clause 8.13.7 describes the usage of MULTI_EXIT_DISC, and Appendix J provides several illustrative examples.

White spaces can be used between symbols to improve readability. ″<>″ denotes the empty sequence.

There are two built-in functions, PathLength() and PathWeight(). PathLength() takes the RD path as an argument and returns the number of RDIs in that path. PathWeight() takes the RD path and an RDI weight table as arguments and returns the sum of weights of the RDIs in the RD path as defined by the RDI weight table. In order to preserve determinism, the RDI weight table must always have a default weight which will be assigned to any RDI which is not in that table.

The RDI path, as used above, is constructed from right to left (which is consistent with IDRP), so that the most recent RD in the path occupies the leftmost position. Each destination NSAP address prefix (and its associated complete RD path) received from other BIS neighbors is matched against local Routeing Policies.

If either no match occurs or the degree of preference associated with the matched policy is REJECT, then the received information is rejected. Otherwise, a degree of preference associated with the matched policy is assigned to that path. Notice that the process terminates on the first successful match. Therefore, policy-terms should be ordered from more specific to more general.

The semantics of a matched policy is as follows: If a destination in <destination-list> that was originally introduced into IDRP from <origin> is received via <RD-path>, that network should be redistributed to all RDs in <distribution-list>.

The interconnection of routeing domains shown in Figure 12 can be used as an example to illustrate how policy terms can be written. In this picture, there are four destination NSAPs, ten BIS's, and five Routeing Domains.

First, consider RD 2. It has no destination NSAPs attached, and models a transit routeing domain, such as a backbone network. It may have a very simple policy: it will carry any traffic between any two RDs, without further constraint. If RD 1 and RD 3 are neighboring domains, then its policy term could be written as:

```
RD 2: < ANY > < (1 | 3) .* > < IDRP > < 1 3 >
= 10
```

The first component in this policy, the destination NSAP list

```
   < ANY >
```

says that any destination is subject to this policy. The second component, the RD path

```
   < (1 | 3) .* >
```

says that routeing information that came from either RD 1 or RD 3 matches this policy, including routes from RDs that lie beyond RD 1 and RD 3. The third component, the origin

```
   < IDRP >
```

says that this route was learned via mechanisms defined in the IDRP protocol. This means that routes learned by other mechanisms mechanisms would not match this policy. The fourth component, the distribution list

```
   < 1  3 >
```

says that this route may be redistributed to both RD 1 and RD 3. Finally, the degree of preference assigned to any route which matches this policy is set to 10.

To improve readability, the above policy can be rewritten as:

```
 RD 2: <ANY><(1|3) ANY*><IDRP><1 3> = 10
```

Next, consider RD 3. It is willing to provide transit service to RD 4 and RD 5, presumably due to multilateral agreements. RD 3 should set its policy as follows:

```
 RD 3: <ANY><(4|5)><IDRP><2 4 5> = 10
 RD 3: <ANY><2 .* ><IDRP><4 5> = 10
 RD 3: <ANY><3><ANY><2 4 5> = 10
```

This would allow RD 3 to distribute internal routes received from RDs 4 and 5 to RDs 2, 4, and 5, and all backbone routes through RD 2 would be distributed to RDs 4 and 5. RD 3 would advertise its own networks to RDs 2, 4, and 5. Destination NSAPs in RD 4 and RD 5 would be able to reach each other, as well as destination NSAPs in RDs 1 and 3 and anything beyond them. RD 3 allows any origin in routes from RD 2. This implies that RD 3 trusts RD 2 to impose policy on routes imported by means other than IDRP. Note that although the policy statement would appear to allow RD 3 to send RDs 4 and 5 their own routes, the IDRP protocol would detect this as a routeing loop and prevent it.

Now consider RD 1. RD 1 wishes to use the backbone service provided by RD 2, and is willing to carry transit traffic for RD 4. The policy statements for RD 1 might read:

```
 RD 1: <ANY><4><IDRP><2>=150
 RD 1: <ANY><2.*><ANY><4> = 150
 RD 1: <ANY><1><ANY><2 4> = 150
```

RD 1 will redistribute all routes learned from the RD 2 backbone to RD 4, and vice versa, and distribute routes to its own networks to both RD 2 and RD 4. The degree of preference assigned to any route which matches this policy is set to 150.

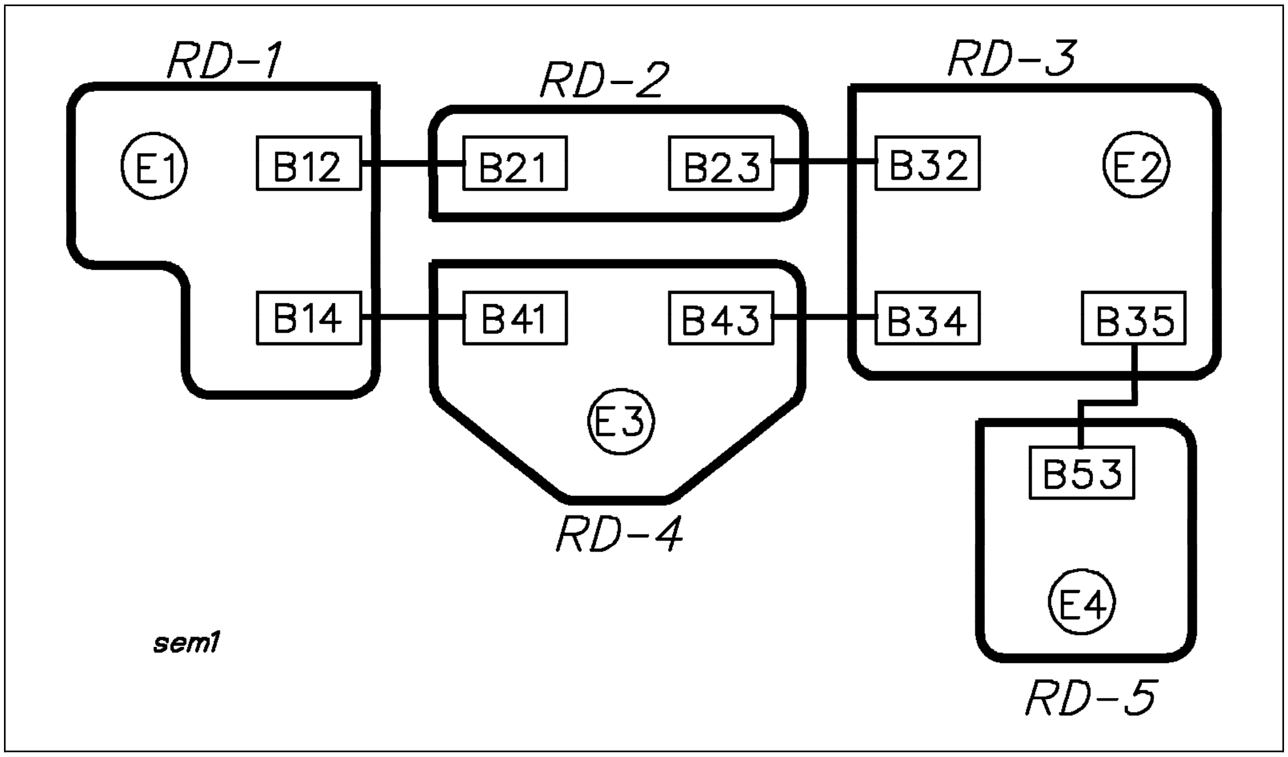

Figure 12. Example Interconnection of Routeing Domains

RD 5 is a more interesting case. RD 5 wishes to use the backbone service, but is not directly connected to RD 2. Its policy statements could be as follows:

```
 RD 5: <ANY><3 4><IDRP><> = 10
 RD 5: <ANY><3 2 .* ><.>< > = 10
 RD 5: <ANY><5><.><3> = 10
```

This policy imports routes through RD 2 and RD 3 into RD 5, and allows RD 5 and RD 4 to communicate through RD 3. Since RD 5 does not redistribute any routes other than its own, it is a stub ERD. Note that RD 5 does not trust RD 3 to advertise only routes through RD 2, and thus applies its own filter to ensure that it only uses the backbone. This lack of trust makes it necessary to add the second policy term.

RD 4 is a good example of a multihomed ERD. RD 4 wishes to use RD 3 as is primary path to the backbone, with RD 1 as a backup. Furthermore, RD 4 does not wish to provide any transit service between RDs 1 and 3. Its policy statement could read:

```
 RD 4: <ANY><3 .*><ANY >< > = 10
 RD 4: <ANY><1 .*><ANY>< > = 20
 RD 4: <ANY><4><ANY><1 3> = 10
```

Paths to any destination through RD 3 are preferred, but RD 1 will be used as a backup if necessary. Note that since RD 4 trusts RD 3 to provide it with reasonable routes, it is not necessary to explicitly import routes from RD 5. Since the redistribution terms are null except for networks within RD 4, RD 4 will never carry any transit traffic.

Given the topology and policies described above, it becomes apparent that two paths of equal preference would be available from RD 2 to any of the destinations in RD 4. Since ties are not allowed, an arbitrary tie-breaking mechanism would come into play (as described above), which might result in less than optimal routes to some destinations. An alternative mechanism that would provide optimal routes while still allowing fallback paths would be to provide destination-by-destination policies in specific cases, and explicit tie-breaking policies for the remaining destinations. For example, the policies for RD 2 could be rewritten as follows:

```
 RD 2: <47:0005:000042><1 .*><IDRP><3> =
10
 RD 2: <47:0005:000042 > <3 .*><IDRP><1> =
20
 RD 2: <.><1 .* ><IDRP><3> = 20
 RD 2: <.><3 .* ><IDRP><1> = 10
```

Paths to destination NSAP prefix 47:0005:000042 through RD 1 would be preferred, with RD 3 as a fallback; paths to all other destinations through RD 3 would be preferred over those through RD 1. Such optimizations may become arbitrarily complex.

There may be other, simpler ways of assigning the degree of preference to an RD path. The simplest way to assign a degree of preference to a particular path is to use the number of RDIs in the RD path as the degree of preference. This approach reflects the heuristic that shorter paths are usually better than longer ones. This policy can be implemented by using the PathLength() built-in function in the following policy statement:

```
 <ANY><.*><ANY><ANY>  =
        PathLength(RDpath)
```

This policy assigns to any network with an arbitrary RD path a degree of preference equal to the number of RDIs in the RD path; it then redistributes this information to all other BIS speakers. As an example, an RD path which traverses three different Routeing Domains will be assigned the degree of preference 3.

Another approach is to assign a certain degree of preference to each individual RD, and then determine the degree of preference of a particular RD path as the sum of the degree of preferences of the RDIs in that path. Note that this approach does not require the assignment of a specific degree of preference to every RDI in the global OSIE. For RDIs with an unknown degree of preference, a default can be used. This policy can be implemented by using the PathWeight() built-in function in the following policy statement:

```
 <ANY><.*><ANY><ANY>
    = PathWeight(RDpath, RDWeightTable)
```

As an example, if Routeing Domains 145 and 55 have 10 and 15 as their weights in the RDWeightTable, and if the default degree of preference in the RDWeightTable is 50, then an RD path that traverses Routeing Domains 145, 164, and 55 will be assigned degree of preference 75.

The above examples demonstrate some of the simple policies that can be implemented with IDRP. In general, very sophisticated policies based on partial or complete RD path discrimination can be written and enforced. It should be emphasized that movement toward more sophisticated policies will require parallel effort in creating more sophisticated tools for policy interaction analysis.

Annex H. Importing Internal Reachability Information

(Informative)

Editor′s Note:

> During the meeting at Sydney, several member bodies expressed an interest in seeing more detail on how the inter-domain protocol (IDRP) and the intra-domain protocol (DIS 10589) can pass information cooperatively between each other.
>
> As a result of the Sydney meetings, revised text will be circulated for both IDRP and for DIS 10589. However, since the editor has not yet had a chance to review the both texts in detail, he has chosen to to delete the text from SC6 N6120 which addressed cooperation between these protocols, with the intention of developing replacement text after both revised documents are available for review.
>
> The editor asks member bodies to consider the question of information exchange between these two protocols, and to submit suggestions as part of their comments on this document. Members should also consider whether this type of material can best be developed as an annex to IDRP, as an annex to DIS 10589, as a standard in its own right, as a Technical Report, etc.

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 con federation 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 13):

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 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 destined to N1, N2, ... Nm will flow through BIS B1, while traffic from BIS A that 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 14):

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.

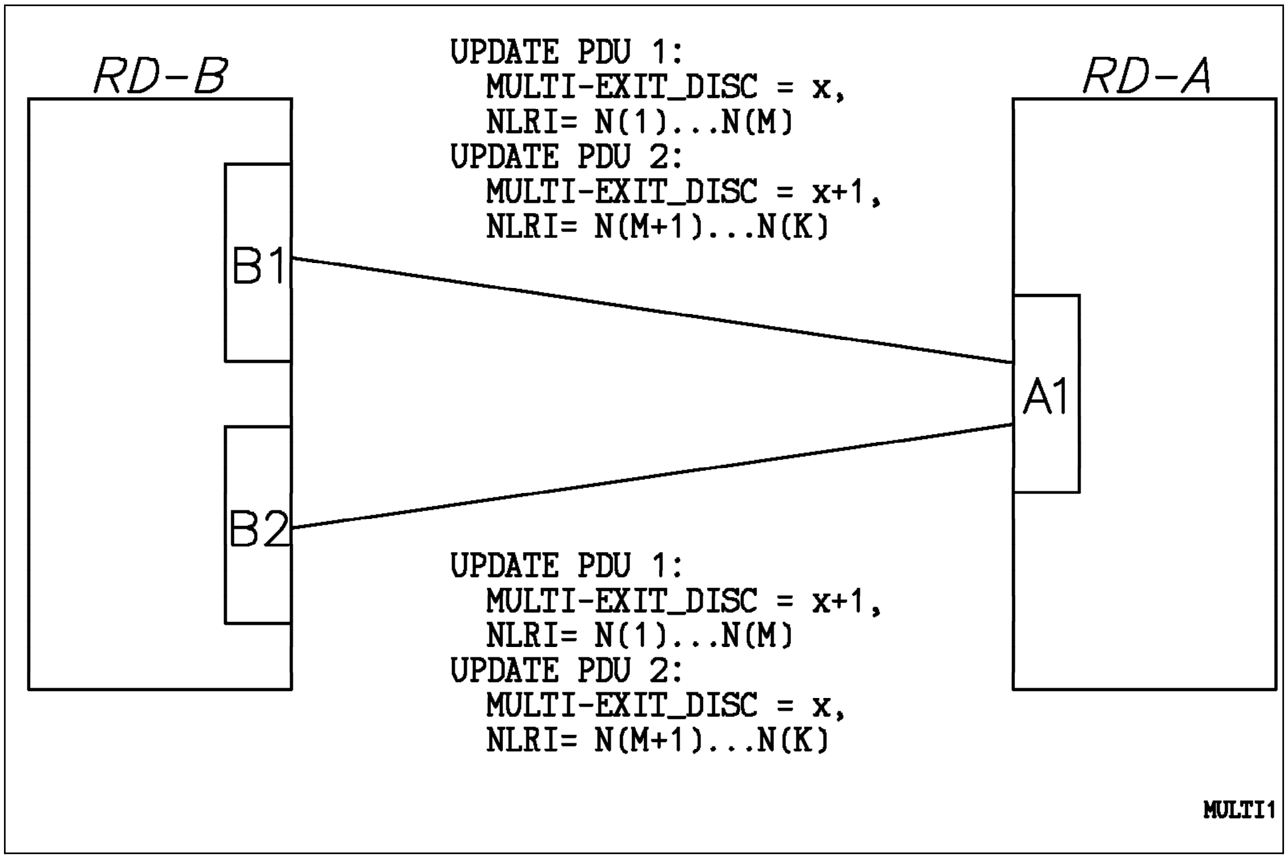

Figure 13. Example 1 Configuration

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 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. Proof of Stability of Route Selection

(Informative)

Table 16. Route Selection Possibilities

Local RD Selection	Adjacent RD Selection
1. Better route	Better route
2. Keep old route	Better route
3. Worse route	Better route
4. Better route	Worse route
5. Keep old route	Worse Route
6. Worse route	Worse route

Given a stable global OSIE interconnect topology, and stable interior information injected by all the RD's of the global OSIE into IDRP, the approach described in 8.11 detects any set of routeing policies (distributed among multiple RD's) that may cause a permanent oscillation of the route selection process in these RD's.

The route selection process may be viewed as a distributed optimization process under constraints. Each RD receives an optimal solution from each of its adjacent RDs. Then, based on its own local objective function, it selects the one that provides local optimum (maximum). As RD's adjacent to the local RD change their optimal solutions (and propagate

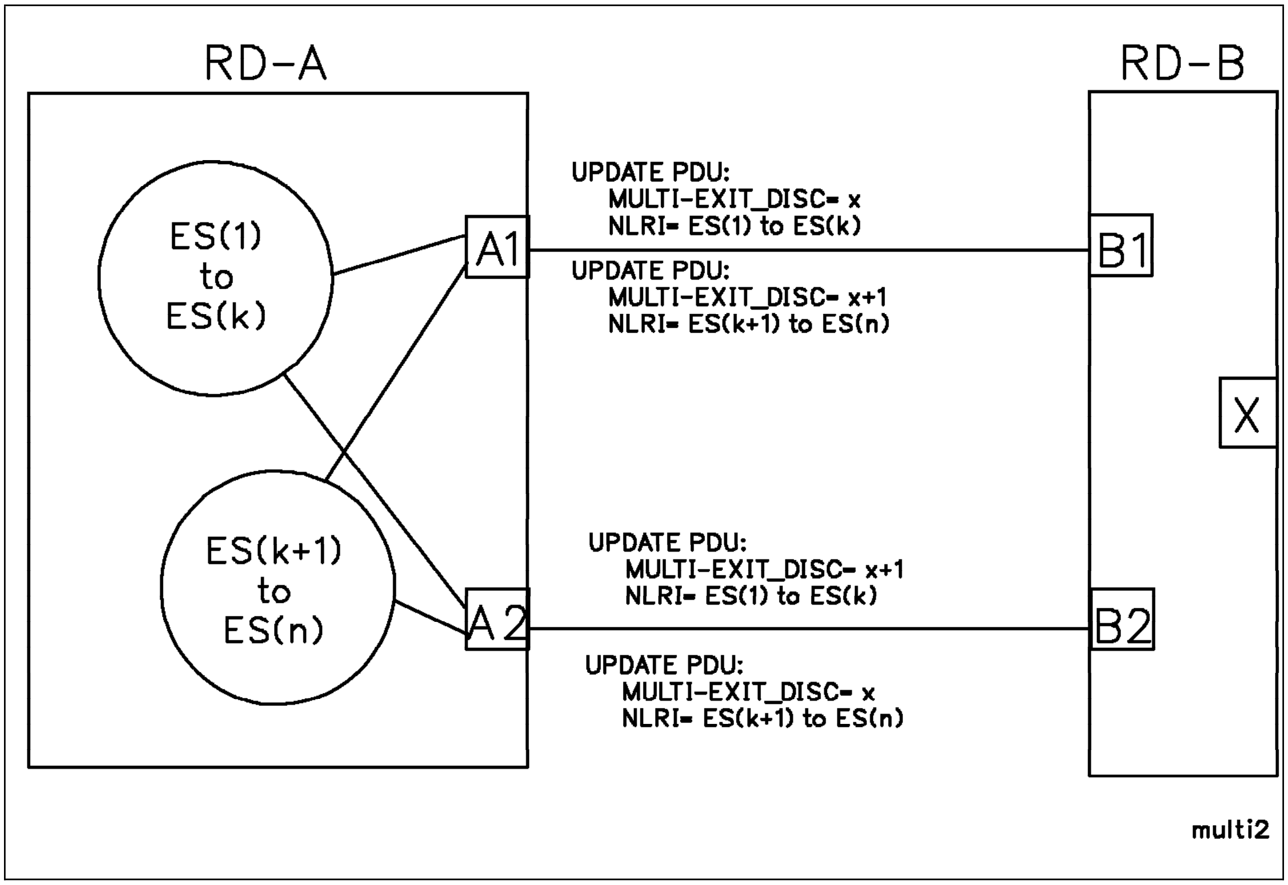
RD-A
RD-B
ES(1) to ES(k)
ES(k+1) to ES(n)
A1
A2
B1
B2
X
UPDATE PDU: MULTI-EXIT_DISC= x NLRI= ES(1) to ES(k)
UPDATE PDU: MULTI-EXIT_DISC= x+1 NLRI= ES(k+1) to ES(n)
UPDATE PDU: MULTI-EXIT_DISC= x+1 NLRI= ES(1) to ES(k)
UPDATE PDU: MULTI-EXIT_DISC= x NLRI= ES(k+1) to ES(n)
multi2

Figure 14. Example 2 Configuration

these solutions to the local RD), the local RD may change its optimal solution as well.

In addition to local RD constraints, there is a global constraint imposed on all of the RD′s in the OSIE—the local optimal solution should not result in formation of a routeing information loop.

To analyze stability of this distributed optimization process, consider situation when the local RD receives an UPDATE_PDU from one of its adjacent RD′s. The adjacent RD may issue a new UPDATE_PDU for an existing route if it either discovers a better route, or the current route becomes unavailable. Similarly, the local RD, as a result of receiving and processing this UPDATE PDU, may either discover a better route, may keep its current route, or may select a route that is worse (from the local RD point of view) than the current one. There are six cases to consider, as shown in Table 16.

Let us define the value of the global OSIE objective function to be equal to the sum of the individual RD′s objective functions. The situation for each of the six possible cases is then as follows:

— Case (1) must eventually stabilize because there are a finite number of possible routes to examine, each RD has a finite set of routeing policies to compare these routes against, it results in the increase of the global OSIE objective function, and the global OSIE objective function has an upper limit. This case may be detected based on the rules (a) and (b) of clause 8.11 and the local route selection process.

— Case (2) must eventually stabilize because there is a finite number of possible routes to examine, the adjacent RD has a finite set of routeing policies to compare these routes against, it results in the increase of the global OSIE objective function, and the global OSIE objective function has an upper limit. This case may be detected based on the rules (a) and (b) of clause 8.11 and the local route selection process.

— Case (3) is the only case where there is a possibility of routeing decision oscillation. If the improvement in the adjacent RD does not depend on the route selected by the local RD, then since there is a finite number of possible routes examine, each RD has a finite set of routeing policies to compare these routes against, and improvement of the adjacent RD is independent of the route selected by the local RD, this process must eventually stabilize. However, if such an improvement depends on the route selected by the local RD, and, as a result of such improvement by the adjacent RD, the local RD must select another route that is worse (as perceived by the local RD) than the current one, this would result in a permanent oscillation of the route selection process in a group of RD′s. This situation may be detected based on the rules (a), (b), (c) of clause 8.11 and the local route selection process.

— Case (4) can never happen since the only reason for the adjacent RD to switch to another route that is worse than the current one is because the current route become unreachable, and according to the rule (1) outlined that results in adjacent RD sending an UPDATE PDU with the UNREACHABLE attribute.

— Case (5) must eventually stabilize because there is a finite number of possible routes to examine, the adjacent RD has a finite set of policies to compare these routes against, it results in the increase of the global OSIE objective function, and the global OSIE objective function has an upper limit. This case may be detected based on the rules (a) and (b) of clause 8.11 and the local route selection process.

— Case (6) must eventually stabilize because there is a finite number of possible routes to examine, each RD has a finite set of policies to compare these routes against, it results in the decrease of the global OSIE objective function, and the global OSIE objective function has a lower limit. This case may be detected based on the rules (a) and (b) of clause 8.11 and the local route selection process.

Annex L. Common Subnetworks

(Informative)

This protocol requires that BISs in adjacent routeing domains be connected by a single subnetwork. For ISO 8802 and point-to-point subnetworks, this can be assured by running ISO 9542. The receipt of a 9542 IS Hello PDU from the remote BIS indicates that the remote BIS is on a common subnetwork.

For ISO 8208 subnetworks, the source and destination DNIC may be compared—if they are equal, the two BISs are likely to share a common subnetwork, although this is not guaranteed.

Note: The issue of what constitutes a subnetwork adjacency within the context of ISO 8208 is left for further study.

Annex M. Example Routeing Algorithm

(Informative)

This annex contains a description of an example routeing algorithm that satisfies the requirements of the protocol described in this international standard:

```
/*
 * ROUTE_SELECT procedure takes the following as the arguments:
 *  1. route R with destination R.dest and N path attributes
 *     R.attrib[1], R.attrib[2], ... , R.attrib[N]
 *  2. K sets of Adj-RIBs-In (one per adjacent BIS)
 *  3. K sets of Adj-RIBs-Out (one per adjacent BIS)
 *  4. M sets of Loc-RIB′s: Loc-RIB[1], Loc-RIB[2], ... , Loc-RIB[M].
 *  5. M sets of PIB′s: PIB[1], PIB[2], ... , PIB[M].
 *  6. M sets of FIB′s: FIB[1], FIB[2], ... , FIB[M].
 *
 * and updates the appropriate Adj_RIB-In, Adj-RIB_Out, Loc_RIB, R,
 * and propagates R to the adjacent BIS′s.
 *
 * It calls function pa2riba() that performs mapping of path
 * attributes to RIB attribute.
 *
 * It calls function rt2dop() that takes route and PIB as an
 * input and return modified route with its degree of preference
 * as the LOCAL_PREF attribute.
 *
 * It calls function find_current_route() that takes Loc_RIB
 * and DESTINATION as arguments and return route to that
 * DESTINATION from the Loc_RIB (if present).
 *
 * It calls function route_feasible() that takes route
 * as an argument and returns TRUE if this route does
 * not have the UNREACHABLE path attribute, and FALSE otherwise.
 *
 * It calls function unfeasible2feasible() that takes
 * route with the UNREACHABLE path attribute and just
 * strips this attribute out
 *
 * It calls function find_new_route() that takes as arguments
 * Adj_RIBS, RIB attribute, destination, and returns route r
 * with the highest degree of preference found in one of Adj_RIBS
 * such that the RIB attribute supplied as an argument is
 * equal to the pa2riba(r).
 *
 * It calls function best_ext() that takes as arguments route, and
 * RIB attribute, and  returns route r to a particular
 * destination that has the highest degree of preference among
 * all other routes to the same destination different from the
 * one supplied as an argument received external inter-domain
 * links.
 *
 * We assume that each route in Adj_RIB′s or Loc_RIB′s has field
 * ″dop″ which is the degree of preference associated with this route.
 */

procedure ROUTE_SELECT(r, adj_ribs_in, adj_ribs_out, loc_ribs, pibs)
INT i, j;
RIB_ATTRIBUTE riba;
ROUTE cr;        /* current route */
ROUTE ext_r;    /* best route received from BIS in an adjacent routeing domain */

switch (route_feasible(r)) {
 case TRUE:      /* feasible route */

  store r in the appropriate adj_ribs;

  for (i = 1; i <= N; i++) {
   riba = pa2riba(r.attrib[i]);
   if (riba == NULL)
    continue;
   for (j = 1; j <= M; j++) {
    if (loc_ribs[j].attrib == riba) {
     cr = find_current_route(loc_ribs[j], r.dest);
     if (cr != NULL) {          /* there are other routes       */
      r = rt2dop(r, pibs[j]);
      if (cr.dop >= r.dop) {    /* no changes to the LOC_RIB    */
       if r was received from a BIS in an adjacent routeing domain {
        ext_r = best_ext(r, riba);
        if (ext_r != NULL && r.dop > ext_r.dop) {
         mark r as ″propagate to internal only″;
         mark r as ″best external″;
         unmark cr as ″best external″;
        }
       }
      }
      else {                    /* install better route      */
       update loc_ribs[j] with r;
       update fibs[j] with r;
       mark r as ″propagate″;
       if r was received from a BIS in an adjacent routeing domain {
        mark r as ″best external″;
        ext_r = best_ext(r, riba);
        if (ext_r != NULL) {
         unmark ext_r as ″best external″;
        }
       }
      }
     }
     else {                     /* no other routes              */
      update loc_ribs[j] with r;
      update fibs[j] with r;
      mark r as ″propagate″;
      mark r as ″best external″;
     }
    }
   }
  }

  update appropriate adj_ribs_out
  propagate all routes marked as ″propagate″;

  propagate all routes marked as ″propagate to internal only″ to
   all other BISs in the same domain;

  break;

 case FALSE:     /* unfeasible route    */

  r = unfeasible2feasible(r);

  remove r from the appropriate adj_ribs_in;
```

```
    for (i = 1; i <= N; i++) {
      riba = pa2riba(r.attrib[i]);
      if (riba == NULL)
        continue;
      for (j = 1; j <= M; j++) {
        if (loc_ribs[j].attrib == riba) {
          cr = find_current_route(loc_ribs[j], r.dest);
          if (cr != r) {        /* no changes to the Loc_RIB    */
            if r was received from a BIS in an adjacent routeing domain {
              if r marked as "best external" {
                update appropriate adj_ribs_out
                propagate r to all other BISs in the domain as unreachable;
                ext_r = best_ext(r, riba);
                if (ext_r != NULL) {
                  mark ext_r as "best external";
                  update appropriate adj_ribs_out
                  propagate ext_r to all other BISs in the domain;
                }
              }
            }
          }
          else {
            remove r from loc_ribs[j];
            update fibs[j];
            update appropriate adj_ribs_out
            propagate r to the adjacent BIS's as unreachable;
            cr = find_new_route(adj_ribs, loc_ribs[j].attrib, r.dest);
            if (cr != NULL) {
              update loc_ribs[j] with cr;
              update fibs[j] with cr;
              update appropriate adj_ribs_out
              propagate cr to the adjacent BIS's;
              if r was received from a BIS in an adjacent routeing domain {
                ext_r = best_ext(r, riba);
                if (ext_r != NULL) {
                  mark ext_r as "best external";
                  if (ext_r != cr) {
                    update appropriate adj_ribs_out
                    propagate ext_r to all other BISs in the domain;
                  }
                }
              }
            }
          }
        }
      }
    }
  }             /* end switch */

  end procedure
```

Annex N. IDRP Network Independent Sublayer Connection Management

(Informative)

This section describes IDRP Network Independent Sublayer connection management operations.

N.1 Connection Record

The variables that define the state of a connection between a pair of BISs are stored in a connection record maintained for each connection. The following describes some of the variables that would be stored in a typical IDRP connection record. It is not intended to be an implementation specification nor is it a complete description. The purpose of naming and describing some of the connection record fields is to simplify the description of the IDRP protocol operation, particularly IDRP service primitives processing.

The connection record fields and their description follow:

— STATE:

  The current state of the connection. Legal values are ESTABLISHED, CLOSED, OPEN-SENT, OPEN-RCVD, and CLOSE-WAIT.

— Send Sequence Number Variables:

  - SND.NXT: The sequence number of the next PDU to be sent.
  - SND.UNA: The sequence number of the oldest unacknowledged PDU.
  - SND.ISS: The initial send sequence number. This is the sequence number that was sent in the OPEN PDU or IDRP ERROR PDU.

— Receive Sequence Number Variables:

  - RCV.CUR: The sequence number of the last PDU received correctly and in sequence.
  - RCV.IRS: The initial receive sequence number. This is the sequence number of the OPEN_PDU or NOTIFICATION_PDU that established this connection.

— Other Variables:

  - CLOSEWAIT: A timer used to time out the CLOSE-WAIT state.
  - HOLD-TIMER: A timer used to time out inactive connections.
  - KEEPALIVE-TIMER: A timer used to indicate liveness of a connection
  - SPDU.MAXSIZE: The largest possible PDU size (in octets) that can legally be sent. This variable is specified by the remote BIS in the OPEN_PDU during connection establishment.
  - SND.MAX: The maximum number of outstanding (unacknowledged) PDUs that can be sent. The sender should not send more than this number of PDUs without getting an acknowledgement. This variable is specified by the remote BIS in the OPEN PDU during connection establishment.
  - RPDU.MAXSIZE: The largest possible PDU size (in octets) that can be received. This variable is specified by the local BIS. The variable is sent to the remote BIS in the OPEN PDU.
  - RCV.MAX: The maximum number of PDUs that can be buffered for this connection. This variable is specified by the local BIS. The variable is sent to the remote BIS in the OPEN PDU.

— Variables from Current PDU:

  - PDU.SEQ: The sequence number of the PDU currently being processed.
  - PDU.ACK: The acknowledgement sequence number of the PDU currently being processed.
  - PDU.MAX: The maximum number of outstanding PDUs the receiver is willing to hold, as specified in the OPEN PDU that established the connection.
  - PDU.MAXSIZE: The maximum PDU size (in octets) accepted by the remote BIS on a connection, as specified in the OPEN PDU that established the connection.

N.2 IDRP Connection Management Pseudocode

This section describes operations of the IDRP Network Independent Sublayer in terms of the IDRP service primitives. The following are the service primitives that are defined by IDRP:

— IDRP requests:

  .
  - START.REQUEST
  - STOP.REQUEST
  - SEND-PDU.REQUEST

— IDRP indications:

  - RCVD-PDU.INDICATION

— IDRP Timeouts:

  - Retransmission Timeout
  - Close-Wait Timeout
  - Keepalive Timeout
  - Hold Time Timeout

The processing of the IDRP service primitives is described by the pseudo-code shown below.

```
START.REQUEST

 CLOSED STATE:

   Create a connection record
   If none available
     Return "Error - insufficient resources"
   Endif

   Generate SND.ISS
   Set SND.NXT = SND.ISS + 1
       SND.UNA = SND.ISS
   Initialize RCV.MAX, RPDU.MAXSIZE
   Initialize HOLD-TIMER Timer
   Send OPEN PDU :
     <Seq=SND.ISS>
     <Ack=0>
     <OutstandingPDUs = RCV.MAX>
     <MaxPDUSize = RPDU.MAXSIZE>
   Set STATE = OPEN-SENT
   Return


 SYN-SENT STATE:
 SYN-RCVD STATE:
 OPEN STATE:
 CLOSE-WAIT STATE:

   Return "FSM Error"
```

```
STOP.REQUEST

 ESTABLISHED STATE:

   Send CEASE PDU :
     <Seq=SND.NXT>
     <Ack=RCV.CUR>
   Set STATE = CLOSE-WAIT
   Start TIMWAIT Timer
   Cancel KEEPALIVE Timer
   Cancel HOLDTIME Timer
   Cancel RETRANSMISSION Timer
   Return

 OPEN-RCVD STATE:

   Send CEASE PDU:
     <Seq=SND.NXT>
     <Ack=RCV.CUR>
   Set STATE = CLOSED
   Return

 OPEN-SENT STATE:

   Send CEASE PDU:
     <Seq=SND.NXT>
     <Ack=0>
   Set STATE = CLOSED
   Return


 CLOSE-WAIT STATE:
 CLOSE STATE

   Return "FSM Error"
```

```
SEND-PDU.REQUEST

  OPEN STATE:

    If PDU type is not UPDATE
       Return "FSM Error"
    Endif

    If PDU Length is greater than SPDU.MAXSIZE
      Return "Error - PDU is too large"

    If PDU type is CEASE or ERROR
      Send PDU:
      Send PDU :
        <Seq=SND.NXT>
        <Ack=RCV.CUR>
      Set SND.NXT = SND.NXT + 1
     Return
    Endif

    If SND.NXT < SND.UNA + SND.MAX
      Send PDU :
        <Seq=SND.NXT>
        <Ack=RCV.CUR>
      Set SND.NXT = SND.NXT + 1
      Restart KEEPALIVE Timer
      Return
     else
      Return "Error - insufficient resources to send data"
    Endif


  SYN-RCVD STATE:
  SYN-SENT STATE:
  CLOSE STATE:
  CLOSE-WAIT STATE:

    Return "FSM Error"
```

The following defines the processing of the arrived BISPDUs from a remote BIS as indicated by the RCVD-PDU.INDICATION service primitive.

```
RCVD-PDU.INDICATION

  CLOSED STATE:

    If CEASE PDU
      Discard PDU
      Return
    Endif

    If OPEN, NOTIFICATION, or UPDATE PDU
      Send CEASE PDU :
        <Seq=0>
        <Ack=PDU.SEQ>
      Discard PDU
      Return
    Endif

    If KEEPALIVE PDU
      Send CEASE PDU :
        <Seq=PDU.ACK + 1>
        <Ack=PDU.SEQ>
      Discard PDU
      Return
    Endif


  CLOSE-WAIT STATE:

    If CEASE PDU
      Set State = CLOSED
      Discard PDU
      Cancel TIMWAIT timer
      Cancel KEEPALIVE Timer
      Cancel HOLDTIME Timer
      Cancel RETRANSMISSION Timer
      Deallocate connection record
      Return
    Endif



    If IDRP ERROR PDU
      Send CEASE PDU :
        <Seq=SND.NXT>
        <Ack=PDU.SEQ>
      Discard PDU
      Return
    Endif

    If OPEN, UPDATE, orKEEPALIVE PDU
      Discard PDU
      Return
    Endif
```

```
OPEN-SENT STATE:

  If PDU.ACK !=SND.ISS
    If PDU.ACK != 0
      Send CEASE_PDU
        <SEQ=PDU.ACK + 1>
    endif
   Deallocate connection record
   Set state = CLOSE-WAIT
  endif

  If CEASE PDU
    Signal "Connection Refused"
    Set STATE = CLOSED
    Discard PDU
    Deallocate connection record
  Endif


  If IDRP ERROR PDU
    Set RCV.CUR = PDU.SEQ
    Send CEASE PDU :
      <Seq=SND.NXT>
      <Ack=RCV.CUR>
    Discard PDU
    Set state = CLOSE-WAIT
    Return
  Endif

  If OPEN PDU
    Set RCV.CUR = PDU.SEQ
        RCV.IRS = PDU.SEQ
        SND.MAX = PDU.MAX
        SPDU.MAXSIZE = PDU.MAXSIZE
    If OPEN PDU successfully passed the local error checking
      If PDU.ACK != 0
        Set SND.UNA = PDU.ACK + 1
          Set STATE = ESTABLISHED
          Send KEEPALIVE PDU :
            <Seq=SND.NXT>
            <Ack=RCV.CUR>
          Start KEEPALIVE Timer
         else
          Set STATE = OPEN-RCVD
          Send OPEN PDU :
            <Seq=SND.ISS>
            <Ack=RCV.CUR>
            <OutstandingPDUs=RCV.MAX>
            <MaxPDUSize=RPDU.MAXSIZE>
        Endif
        Return
      else
        If required to send IDRP ERROR PDU
          If PDU.ACK != 0
            Set SND.UNA = PDU.ACK + 1
            Send IDRP ERROR PDU :
              <Seq=SND.NXT>
              <Ack=RCV.CUR>
          else
            Send IDRP ERROR PDU :
              <Seq=SND.ISS>
              <Ack=RCV.CUR>
          Set SND.NXT = SND.NXT + 1
          Endif
          Set STATE = CLOSE-WAIT
        Endif
      Endif
    Endif


    If anything else
      Discard PDU
      Return
    Endif
```

```
OPEN-RCVD STATE:

  If RCV.IRS < PDU.SEQ =<RCV.CUR+RCV.MAX
    PDU sequence number acceptable
  else
    Send KEEPALIVE PDU :
      <Seq=SND.NXT>
      <Ack=RCV.CUR>
    Discard PDU
    Return
  Endif


  If CEASE PDU
    Discard PDU
    Return
  Endif

  If OPEN PDU
    Send CEASE PDU :
      <Seq=PDU.ACK + 1>
    Set STATE = CLOSED-WAIT
    Signal "Connection Reset"
    Discard PDU
    Deallocate connection record
    Return
  Endif

  If KEEPALIVE PDU
    If PDU.ACK = SND.ISS
        Set STATE = ESTABLISHED
      else
        Send CEASE PDU :
          <Seq=PDU.ACK + 1>
          <Ack=0>
        Set STATE = CLOSE-WAIT
        Discard PDU
        Return
    Endif
  Endif


  If IDRP ERROR PDU
    If PDU.ACK = SND.ISS
        Send CEASE PDU :
          <Seq=SND.NXT>
          <Ack=PDU.SEQ>
        Set STATE = CLOSE-WAIT
        Return
    else
        Discard PDU
        Return
  Endif

If UPDATE PDU
  If the received segment is in sequence
    Check PDU for correctness
    If erroneous PDU
      If IDRP ERROR PDU should be sent
        Set RCV.CUR = PDU.SEQ
        Send IDRP ERROR PDU :
          <Seq = SND.NXT>
          <Ack = RCV.CUR>
        Set STATE = CLOSE-WAIT
        Return
      else
        Discard PDU
        Return
      Endif
    else
      Restart HOLD-TIME Timer
      Update proper ADJ-RIB
      Set RCV.CUR = PDU.SEQ
      Send KEEPALIVE PDU :
        <Seq = SND.NXT>
        <Ack = RCV.CUR>
      Start KEEPALIVE Timer
      Set STATE = ESTABLISHED
      Return
   Endif
 else
   Discard PDU
   Return
Endif
```

```
ESTABLISHED STATE:

 If IDRP ERROR PDU
   Set RCV.CUR=PDU.SEQ
   Send CEASE PDU :
     <Seq=SND.NXT>
     <Ack=RCV.CUR>
   Set STATE = CLOSE-WAIT
   Cancel HOLD-TIME Timer
   Cancel KEEPALIVE Timer
   Return
 Endif

 If CEASE PDU
   Set STATE = CLOSE-WAIT
   Discard PDU
   Signal "Connection Reset"
   Return
 Endif

 If RCV.CUR < PDU.SEQ =< RCV.CUR + RCV.MAX
   PDU sequence number acceptable
  else
   Send KEEPALIVE PDU :
     <Seq=SND.NXT>
     <Ack=RCV.CUR>
   Restart KEEPALIVE Timer
   Discard PDU
   Return
 Endif

 If OPEN PDU
   Signal "Connection Reset"
   Discard PDU
   Return
 Endif

 If KEEPALIVE PDU
   If SND.UNA =< PDU.ACK < SND.NXT
     Set SND.UNA = PDU.ACK + 1
     Flush acknowledged PDUs
   Endif
   Restart HOLD-TIME Timer
 Endif


 If UPDATE PDU
   If the received PDU is in sequence
     Check PDU for correctness
     If erroneous PDU
       If IDRP ERROR PDU should be sent
         Set RCV.CUR = PDU.SEQ
         Send IDRP ERROR PDU :
           <Seq = SND.NXT>
           <Ack = RCV.CUR>
         Set STATE = CLOSE-WAIT
         Set SND.NXT = SND.NXT + 1
         Return
       else
         Discard PDU
         Return
       Endif
     else
       Restart HOLD-TIME Timer
       Update proper ADJ-RIB
       Set RCV.CUR = PDU.SEQ
       Send KEEPALIVE PDU :
         <Seq = SND.NXT>
         <ACK = RCV.CUR>
       Restart KEEPALIVE Timer
       Return
     Endif
  else
    Discard PDU
    Return
 Endif
```

Timeout events occur when a timer expires and signals the IDRP. Four types of timeout events can occur, as described below:

RETRANSMISSION Timeout

```
If timeout on PDU at head of retransmission queue
   Resend the PDU at head of queue
   Restart the retransmission timer for the PDU
   Requeue the PDU on retransmission queue
   Return
Endif
```

CLOSE-WAIT Timeout

```
Set STATE = CLOSED
Deallocate connection record
Return
```

KEEPALIVE Timeout

```
If STATE != ESTABLISHED
  Return "FSM Error"
else
  Send KEEPALIVE PDU:
    <Seq = SEQ.CUR>
    <Ack = RCV.CUR>
  Restart KEEPALIVE Timer
Endif
Return
```

HOLD-TIME Timer Timeout

```
Mark all routes received from the remote BIS
as unreachable.
Update Loc-RIBs.
Set STATE = CLOSED
Deallocate connection record
Return
```

Index