**Title:** Ballot Comments on DIS 10747 (SC6 N7692)

**Source:** X3S3.3

X3S3.3 recommends a vote of ________ with the following comments, as shown in the attachment.

This contribution is a consolidation of several contributions that we discussed at our November 1992 meeting. In those cases where the original contributions did not classify the comment (Major Technical, Minor Technical, Major Editorial, or Minor Editorial), I took the liberty of doing so.

In my view, there are four major comments:

1. Addition of Globally Unique Security Attribute
2. Carrying Multiple Routes in a Single UPDATE PDU
3. Weak/strong Forwarding Mechanisms
4. Rapid BIS-BIS Connection Close

If we agree that they are "Major Technical", then we must vote for DISAPPROVAL; if not, then we can vote "APPROVE with COMMENTS". It's up to us to fill in the blank (above) appropriately.

Noting that the prospective USA comments will run about 35 pages in length, and noting that several of them do major surgery to the FSMs and the path attributes (in practice, although from an abstract sense they are all "straightforward"), I plan to propose the following options at the London meeting:

1. If all NBs reach technical consensus and all NBs in fact submit their "final" ballot comments (which are not actually due until April 22, 1993), I will propose hosting an editing meeting for IDRP in May or June to allow interested experts to proofread the revised text before submission to the ITTF.

2. If NBs do not submit final comments or do not reach technical consensus, then I will revise the document, as appropriate, after having received all ballot comments, and will plan to discuss the revised text at the SC6/WG2 meeting in September 1993.

**Note to Reviewers**

The IDRP discussions are scheduled for the first afternoon of our January meeting. Please try to read this set of consolidated comments in advance, so that we can make the best use of our time during the meeting. Remember, due to the various backups built in to the ANSI review cycles, we must produce our final recommended ballot comments at the January meeting.

***Major Technical Comments***

1. **Globally Unique Security, several places in text (Major Technical):**

Although ISO 8473 defines a Globally Unique Security option, IDRP provides no support for it. According to the current text of 8.2, IDRP does not treat Globally Unique Security as an NPDU-derived Distinguishing Attribute: hence, even if present, this parameter plays no role in the selection of an entry in the Forwarding Information Base.

Since there are networks under design that depend on the use of this attribute in an inter-domain environment, IDRP should support this option. The mechanics of support are relatively straightforward since this parameter is similar to other already-supported ISO 8473 parameters.

The changes needed occur throughout the document:

- Add a new item ″r″ to clause 6.3, renumbering other type codes as appropriate:

  *GLOBALLY UNIQUE SECURITY (Type Code 18)*

  *This is a well-known discretionary attribute whose variable length field contains a security label for use in conjunction with ISO 8473′s globally unique security parameter; it is encoded as follows:*

  | |
  |---|
  | Length (1 octet) |
  | Security Label (variable) |

  1. *Length:*

     *Gives the length in octets of the Security Label field*

  2. *Security Label:*

     *Contains the value of the security label*

  *Its usage is defined in (new) 7.12.18.*

- Create a new clause 7.12.18, with title ″Globally Unique Security″, and renumber remaining clauses appropriately. The text of the new clause is as follows:

  *GLOBALLY UNIQUE SECURITY is a well-known discretionary attribute that allows a BIS to specify the security label that is associated with a given route, thus preventing use of that route by NPDUs that do not contain the specified security label in the globally unique security parameter.*

  *The BIS that originates a route may optionally set the GLOBALLY UNIQUE SECURITY path attribute in accordance with its security policies. A BIS that redistributes a route containing the GLOBALLY UNIQUE SECURITY path attribute shall not modify or delete this parameter.*

- Update clause 8.2 (second dashed item) to show that Globally Unique Security is indicated by the value ″1″ in each of the first two bits of the ISO 8473 security parameter.
- Include GLOBALLY UNIQUE SECURITY as a distinguishing path attribute in Table 3.
- Add GLOBALLY UNIQUE SECURITY to the list given in item ″b″ of clause 7.11.2 on page 36.

- Add GLOBALLY UNIQUE SECURITY to the list of type-value specific attributes contained in clauses 7.11.3 and 8.3
- Add new text, shown below, to the end of 8.3 to define the criteria for a match with respect to the globally unique security parameter:

  *For the Globally Unique Security parameter, a match occurs when the security label carried in the NPDU-derived distinguishing attribute has the same value as the security label carried in the FIB-Att.*
- Generate appropriate GDMO and ASN.1 for this new path attribute.
- Update PICs and conformance sections in accordance with the previous description of this attribute, showing that this is an optional path attribute.

2. **Carriage of Multiple Routes in a Single UPDATE PDU (Major Technical):**

The DIS text for IDRP permits a single UPDATE PDU to advertise only a single route. The RIB associated with that route is identified by a unique RIB-Att derived from the distinguishing path attributes carried in that UPDATE PDU. While this approach is suitable for networks that are not bandwidth-constrained, it is not suitable for implementing IDRP over low bandwidth networks. In such situations, it can be advantageous to carry several routes within a single UPDATE PDU.

Although the current mechanisms of one UPDATE PDU per RIB-Att allows a simple mapping from the UPDATE PDU into the associated RIB, it may require a significant amount of time for the RIBs to stabilize when multiple routes must be exchanged between a pair of BISs. Furthermore, if the topology of interest contains many routeing domains, it could take a long time for the total topology to stabilize on the correct, up-to-date routeing information.

We suggest that relatively simple changes can be made to permit a single UPDATE PDU to carry multiple routes, to associate a RIB with each such route, and to provide an unambiguous RIB-ID for for each such route. This method will allow more efficient encoding of the UPDATE PDU in cases where the non-distinguishing attributes of a given link are the same for multiple sets of distinguishing attributes (that is, for RIB-Atts). An example of this type of situation would be a single subnetwork connecting a pair of BISs, where a value of two distinguishing path attributes (say TRANSIT DELAY and EXPENSE) are associated with the same physical path. The proposed method would allow a single UPDATE PDU to report both of these routes: one associated with the EXPENSE RIB and the other with the DELAY RIB.

In outline, the method is as follows:

a. Associate a numerical RIB-ID with each unique set of distinguishing attributes (that is, each unique RIB-Att) that a given BIS supports. The identifying number, called the RIB-ID, is derived unambiguously from the position of the RIB-Att in the OPEN PDU.

b. Change the name of ROUTE-ID to ROUTE-LIST, and expand the current definition so that it contains a set of 3-tuples of the form <ROUTE-ID, RIB-ID, LOCAL_PREF>. This will associate a distinct route-identifier and a local preference value (where appropriate) with each of the (multiple) routes carried in an UPDATE PDU.

c. Delete the existing LOCAL PREF attribute, since this information is now going to be carried in the new ROUTE-LIST

d. A BIS that receives an UPDATE PDU will expand the routeing information into multiple routes: the new ROUTE-LIST gives the RIB-Att and the ROUTE-ID for each of the routes carried.

e. If an UPDATE PDU has a RIB-ID that does not match with the actual distinguishing path attributes in the UPDATE PDU (for example, RIB-ID says that EXPENSE is supported, but the BISPDU contains no EXPENSE attribute), then the UPDATE PDU will be discarded.

No changes are needed for withdrawing routes, since each route has a unique route identifier, whether advertised in a single UPDATE PDU or in an UPDATE PDU that advertises multiple routes.

The changes needed to accomplish this are as follows:

a. Insert a new last paragraph in the description of **RIB-ATTSet** on page 13, immediately in front of the note:

   *The order in which the RIB-Atts are listed is used to generate the values for the RIB-ID field of the ROUTE_LIST path attribute, as described in clause 7.12.1.*

b. Clause 5.8, page 8: Replace the second paragraph of this clause with the follows:

   *When a BIS receives an UPDATE PDU, it examines the ROUTE_LIST path attribute to determine how many routes are being advertised in that UPDATE PDU. The ROUTE_LIST associates each route with a RIB-Att, and the RIB-Att in turn identifies the routeing information base for that route.*

c. Clause 6.3, page 13: To reflect the capability of carrying multiple routes in a single UPDATE PDU, the following changes are needed:

   - In the first paragraph, change ″A single feasible route″ to ″feasible routes″ in the first sentence; and change ″a feasible route″ to ″multiple feasible routes″.
   - In the second list item, change ″feasible route is″ to ″feasible routes are″
   - In the second paragraph, change ″at most one route″ to ″multiple routes″.
   - In the fourth paragraph, change ″a feasible route″ to ″feasible routes″

d. Clause 6.3, page 15, item ″a″: Replace the existing text with the following:

   *ROUTE_LIST (Type Code 1):*

   *ROUTE-LIST is a well-known mandatory attribute that provides a ROUTE-ID, RIB-ID, and LOCAL-PREF value for each route carried in the UPDATE PDU. It can contain multiple 3-tuples when the UPDATE PDU advertises multiple routes. It is encoded as shown below:*

| |
|---|
| ROUTE COUNT (1 octet) |
| ROUTE-ID 1 (4 octets) |
| RIB-ID 1 (1 octet) |
| LOCAL-PREF 1 (1 octet) |
| ROUTE-ID 2 (4 octets) |
| RIB-ID 2 (1 octet) |
| LOCAL-PREF 2 (1 octet) |
| ... |
| ROUTE-ID n (4 octets) |
| RIB-ID n (1 octet) |
| LOCAL-PREF 2 (1 octet) |

1. *ROUTE COUNT:*

   *A positive integer that gives the number of <ROUTE-ID, RIB-ID, LOCAL-PREF> 3-tuples that follow. There will be one 3-tuple for each route that the UPDATE PDU is advertising. A count of zero indicates that no routes are being advertised.*

2. *ROUTE-ID:*

   *A four octet field that contains the route identifier for the route associated with the RIB-Att identified by the RIB-ID in the immediately following field.*

3. *RIB-ID:*

   *A one octet numerical identifier for the RIB with which this route is associated.*

4. *LOCAL_PREF: A one octet field that contains the local preference value for route.*

*The usage of this attribute is defined in clause 7.12.1.*

e. Clause 7.12.1, page 37: Change the clause heading to "ROUTE-LIST", delete the existing two paragraphs of text, and replace it with the following new text:

   *ROUTE-LIST is a well-known mandatory attribute. The ROUTE COUNT shall be set to the number of feasible routes that are being advertised in this UPDATE PDU. A ROUTE COUNT of zero indicates that no feasible routes are being advertised.*

   *For a ROUTE COUNT of n, there will be exactly n 3-tuples <ROUTE-ID, RIB_ID, LOCAL-PREF> —that is, one 3-tuple for each route. A BIS shall include a 3-tuple for each feasible route carried in the UPDATE PDU:*

   - *The ROUTE-ID must be unambiguous within the context of the BIS-BIS connection over which the UPDATE PDU is transmitted.*
   - *The RIB-ID identifies the distinguishing attributes of the route: that is, it identifies a RIB-Att. The RIB-ID for a given RIB-Att shall be numerically equal to the position of the RIB-Att in the local BIS's OPEN PDU. That is, the first listed RIB-Att shall be assigned a RIB-ID of 1, the second listed shall assigned a RIB-ID of 2, etc. The Empty RIB-Att shall be assigned a RIB-ID of 0.*

- *The LOCAL-PREF field is used to detect inconsistent routeing decisions among a set of BISs that are all located in the same routeing domain. Its value shall be set as follows:*

  1. *For UPDATE PDUs sent to adjacent routeing domains, LOCAL-PREF shall contain the value 0; the receiving BIS (in the adjacent RD) shall ignore this field upon receipt.*
  2. *For UPDATE PDUS sent to BISs in the same routeing domain as the local BIS, its value shall be set in accordance with 7.15.1; the receiving BIS (in the same RD) shall use this value to check for internal inconsistencies, in accordance with 7.15.1.*

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

f. In Note 24 on page 37, delete item ″a″: since multiple routes can be carried in an UPDATE PDU, this item is no longer correct.

   In item ″b″, change ″Since ROUTE-ID must be unambiguous...″ to ″Since ROUTE-ID and RIB-ID must only be unambiguous...″; change ″using different ROUTE-IDs″ to ″using different values for these fields...″

   Item ″c″ should be deleted, since it is redundant with the last paragraph in the proposed new text above.

g. Clause 7.14, item ″d″, page 47: The language needs to be adjusted to reflect the fact that an UPDATE PDU could advertise several feasible routes. Therefore, the following changes are necessary:

   - Replace the first paragraph of ″d″ with:

     *When an UPDATE PDU is received, the BIS shall update the appropriate Adj-RIBs-In. For each feasible route, the Adj-RIB-In is identified by the RIB-ID carried in the ROUTE-LIST parameter of the UPDATE PDU; for each unfeasible route, the route is identified by the ROUTE-ID carried in the WITHDRAWN ROUTES field of the UPDATE PDU. The actions to be taken for each route are:*

   - In item ″d)2)″, replace the first paragraph with:

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

h. Clause 7.15.1, page 47: Change ″LOCAL_PREF attribute″ to ″LOCAL_PREF field of the ROUTE_LIST attribute″ (2 places).

   Change the last sentence of the first paragraph to read:

   *Its value shall be set to zero in UPDATE PDUs transmitted to BISs that are located in adjacent routeing domains. For routes advertised to BISs located in the same routeing domain as the local BIS, its value shall be set to the degree of preference for the route as computed by the advertising BIS.*

Delete the second paragraph in its entirety ("The following procedures shall be applied..."). It is superfluous with the immediately following paragraph.

i. Modify item "h" in clause 7.21.3 as follows:

*If any non-empty set of distinguishing attributes in an UPDATE PDU, as described by its RIB-ID fields, does not match any of the RIB-Atts...*

j. Add a new item "o" at the end of clause 7.21.3 (UPDATE PDU Error Handling):

*If, for each RIB-ID in the ROUTE_LIST, the corresponding distinguishing path attributes are not present in the UPDATE PDU, then the UPDATE PDU shall be discarded. The receiving BIS shall send an IDRP ERROR PDU that reports an error subcode of Malformed_Attribute_List.*

3. **Forwarding ISO 8473 NPDUs with Type 2 and 3 Functions (Major Technical):**

The USA believes that the forwarding process of IDRP should be amended to be consistent with the classifications of functions given in ISO 8473.

Within ISO 8473, Security is a type 2 function. Therefore, if the NPDU-derived Distinguishing attributes contain GLOBAL SECURITY, SOURCE SPECIFIC SECURITY, or DESTINATION SPECIFIC SECURITY, and there is no forwarding information base whose RIB-Att contains an equivalent distinguishing path attribute, then the NPDU should be discarded and an 8473 ER PDU should be generated.

However, in ISO 8473, both QOS and Priority are type 3 functions. For NPDUs containing only Type 2 functions, lack of an equivalent RIB-Att should result in their being forwarded along the default path associated with the Empty RIB-Att, rather than their being discarded. The USA notes, however, that "weak forwarding" along the path identified by the Empty RIB-Att can compromise the transit policies of intermediate routeing domains. This occurs because of a deficiency in ISO 8473: namely, its header includes no fields to indicate whether forwarding is "strong" (that is, the entire path supported the requested Type 3 attribute) or "weak" (at some point, the NPDU travelled along the default path).

Notwithstanding this problem with ISO 8473, the USA still believes it advisable to amend IDRP to require strong forwarding for Type 2 parameters, and to allow the option of weak forwarding for Type 3 parameters, including an informative note alerting readers to the potential problem in ISO 8473 if this is done.

We suggest the replacement of items 2, subitems 'i" and "ii", in clause 8 on page 59 with the following new text:

*2) It shall next apply the procedures of clause 8.3 to determine if the NPDU-derived Distinguishing Attributes match the FIB-Atts of the Forwarding Information Bases of the local BIS. The following procedure shall be invoked, in the order indicated:*

1. *If the NPDU-derived Distinguishing Attributes match the FIB-Att of a local FIB, then the NPDU shall select that FIB and shall forward the NPDU using the methods of clause 8.4.*
2. *If the NPDU-derived Distinguishing Attributes do not match any FIB-Att, then the following precedence shall be used to determine an appropriate* <u>*fallback FIB*</u>*, or to discard the NPDU:*
   a. *If the NPDU-derived Distinguishing Attributes include GLOBAL SECURITY, SOURCE SPECIFIC SECURITY, or DESTINATION SPECIFIC SECURITY, and there is no FIB that includes the corresponding distinguishing path attribute in its FIB-Att, 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".*

*If there is at least one FIB with a FIB-Att that includes the corresponding distinguishing security path attribute, then the BIS shall either:*

- *select any such FIB as the fallback FIB, and forward the NPDU using the methods of clause 8.4, or*
- *perform the ISO 8473 "Discard PDU Function" and generate an ER PDU with the parameter value set to "Unsupported Option not Specified".*

*The decision to discard the NPDU or to forward it using a specific fallback FIB chosen from the set of all FIBs that include the corresponding distinguishing path attribute is a local matter.*

b. *If the NPDU-derived Distinguishing Attributes do not include GLOBAL SECURITY, SOURCE SPECIFIC SECURITY, or DESTINATION SPECIFIC SECURITY, and do include EXPENSE, TRANSIT DELAY, RESIDUAL ERROR, CAPACITY, or PRIORITY, then the BIS shall either:*

   - *perform the ISO 8473 "Discard PDU Function" and generate an ER PDU with the parameter value set to "Unsupported Option not Specified", or*
   - *select the FIB that corresponds to the Empty FIB-Att as the fallback FIB, and forward the NPDU using the methods of clause 8.4.*

*NOTE __: In a stable topology, the decision to discard the NPDU or to use a fallback FIB fallback FIB will be made by the exit BIS of the routeing domain that contains the system that generated the NPDU. If a fallback FIB is used, the BIS must insure that the NPDU will not later be forwarded according to a FIB whose FIB-Att is different from that of the fallback FIB. The means to accomplish this are outside the scope of this international standard.*

4. **Rapid Connection Close:**

The FSM defined in the DIS text inserts a long delay during the process of closing a BIS-BIS connection, and there is currently no way provided to avoid it. It would be desirable for IDRP to provide a "rapid close" feature. Such a feature would be useful, for example, when it is necessary to re-establish a BIS-BIS connection to include new information in the OPEN PDU, such as change of confederation membership or support for new RIB-Atts.

The "rapid close" function can be accommodated without adding additional states to the current FSM:

a. An acknowledgement should be added to the close process: that is, a BIS that receives either a CEASE or IDRP ERROR PDU should acknowledge that PDU by sending a CEASE PDU back to its peer BIS.
b. Upon receipt of such an acknowledgement, the BIS that originated the CEASE or the IDRP ERROR PDU should enter the CLOSED state immediately, without entering the CLOSE-WAIT state.
c. If any BISPDU other than a CEASE is received by a BIS that is in the CLOSE-WAIT state, then the BIS should respond with a CEASE PDU.
d. Cease PDUs should be sequenced—that is, a BIS that is in the ESTABLISHED state will discard out-of-sequence CEASE PDUs.

If the initial CEASE or IDRP ERROR PDU from BIS-A is lost, then there are two possibilities:

- If BIS-A receives a CEASE PDU from BIS-B, then the connection will close normally.
- If BIS-A receives any BISPDU other than a CEASE (either because BIS-B never received A′s CEASE, or because a BISPDU from B was already in transit when A sent its CEASE), then BIS-A will reply with a second CEASE PDU and will immediately enter the CLOSED state.
- If BIS B receives the first or the second CEASE, it close the connection normally.
- If BIS-B receives nothing from BIS-A, eventually its Hold Timer will expire.
- If BIS-A attempts to re-establish the connection while B is still in the ESTABLISHED state, the receipt of the OPEN PDU will cause BIS-B to declare an FSM error. BIS-B will then enter the CLOSE-WAIT, leading to an eventual connection close.

If, on the other hand, BIS-B receives the first CEASE from A, but its responding CEASE gets lost in transit, then A (who is in the CLOSE-WAIT state) will eventually time out if it receives no further BISPDUs from B. If it does receive a BISPDU from B, then A will immediately close the connection. (This includes the case where BIS-B attempts to re-open the connection while A is still in the CLOSE-WAIT state.)

There is a potential problem with delayed duplicate CEASE PDUs′ arriving and perturbing a new connection. Delayed duplicate CEASE PDUs are a very low probability event due to the fact that CEASE PDUs are only transmitted in response to received packets (and never retransmitted due to timer expirations) and due to the resynchronization properties of the open and close procedures. However, to further guard against this eventuality, CEASE PDUs should be considered to be sequenced, and out-of-sequence CEASE PDUs should be discarded in the ESTABLISHED state. Since sequence numbers are guaranteed to be monotonically increasing when less than CLOSE_WAIT_DELAY seconds separates two connections, delayed duplicate CEASE PDUs will be ignored when the new connection is established.

The following are the necessary changes:

- Include the CEASE PDU in the list of sequenced BISPDU types in clauses 7.5.3a and 7.5.3c.
- Replace 7.6.1.2j, 7.6.1.3g, and 7.6.1.4f with the following text:

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

- Replace 7.6.1.5b with the following text:

  *b) If the BIS receives an in-sequence CEASE PDU, the FSM shall enter the CLOSED state.*

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

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

  - *Start event*
  - *Stop Event*
  - *Expiration of Hold Timer*

- Change ″CLOSE-WAIT″ to ″CLOSED″ in 7.6.2c and 7.6.2d
- Modify Table 2 as follows:
  - CLOSE-WAIT state, all PDU receptions except CEASE: S=CLOSE-WAIT, A=send CEASE PDU, restart CloseWaitDelay timer

- Receive CEASE PDU, all states except CLOSE-WAIT and CLOSED: S=CLOSED, A=send CEASE PDU
- Receive CEASE PDU, CLOSE-WAIT state: S=CLOSED, A=none

***Minor Technical Comments***

5. Clause 6.3, page 14 (Minor Technical): In considering the encoding for the *Flag* and *Length* the USA notes that there are 5 unused bit positions in the *Flag* field. We also note that the *Length* field can accommodate a path attribute whose maximum length is 65,536 octets, which is larger than is necessary.

   To minimize the number of octets carried in an UPDATE PDU, we suggest consolidating these two fields into a single field of fixed length 2 octets: 3 bits will be allocated to the three existing flags, and 13 bits will be allocated to denote the length of the path attribute (that is, a maximum length of 8,192 octets per attribute.

   A length of 8 K octets can easily accommodate the maxium RD path; 256 RDs * 20 octets is only 5120 octets. Similar analyses would apply to DIST_LIST_INCL and DIST_LIST_EXCL. NEXT_HOP could accommodate approximately a box with about 400 SNPAs. That is, 8K octets is sufficient for all practical cases.

6. Clause 6.3, page 20 (Minor Technical): The NLRI field is overspecified; in particular, the encoding for the Addr_Info field should be specified in a way that does not constrain encoding for protocols other than ISO 8473.

   To accomplish this, replace the first paragraph on Addr_Info with the following:

   > *This field contains a list of reachable address prefixes. The encoding of this field is specific to each protocol supported.*
   >
   > *For use with ISO 8473, this field shall be encoded as one or more 2-tuples of the form <length, prefix>, whose fields are described below:*

   (The remainder of the text for Addr_Info remains unchanged.)

7. Clause 7.3, page 23, item ″d″ (Minor Technical): The description of **INTERNAL-SYSTEMS** as a ″list of the systems contained within the routeing domain″ does not address the form that the list can be expressed in; on the other hand, the associated ASN.1 entry for ″SystemIdGroup″ on page 79 indicates that the list should be constructed from complete NSAPs or NETs.

   This seems to be unnecessarily restrictive: since the information in **INTERNAL-SYSTEMS** is used to construct NLRI, which itself is expressed as prefixes, it would be more natural to use prefixes to express the contents of **INTERNAL-SYSTEMS**. Since a complete NSAP or NET is in fact only a special case of a prefix,. this does not preclude the use of full NSAPs or NETs, if desired.

   The following changes should be made:

   - Item ″d″, clause 7.3: Replace the clause ″which is the list...″ with: ″*which lists the address prefixes of the systems contained within the routeing domain*″.
   - Amend the associated ASN.1 definitions in clause 11.9 as follows:
     ```
       Add:     NETPrefix ::=NSAPprefix
                ESPrefix  ::=NSAPprefix

     Replace:  SystemidGroup ::= SEQUENCE{
                           nETS SET OF NETPrefix,
                           nSAPS SET OF ESPrefix }
     ```

8. Clause 7.5.2, page 25 (Minor technical): The procedures used to mange the sequence number space are both overconstrained and underspecified. In particular, it would be safe to re-use the sequence number space if a least CloseWaitDelay seconds had passed since the last BISPDU was

issued by the local BIS. Conversely, there is no text stating this constraint, which can fail to be met if a machine loses all state (for example, it is rebooted).

To correct this, we recommend the following:

a. Add the following text to the end of 7.5.2a:

> *Before attempting to establish a BIS-BIS connection with an adjacent BIS, the local BIS must ensure that it has not sent a BISPDU to the adjacent BIS for at least* **CloseWaitDelay** *seconds.*

b. Replace 7.5.2c with the following text:

> *If the connection is subsequently closed under the conditions described in Table 2 and a subsequent connection is to be made to the same adjacent BIS, the local BIS shall, as a local matter, choose one of the following options:*
>
> - *Maintain status of the sequence number space, and use any value greater than the last-used value, or*
> - *Ensure that at least* **CloseWaitDelay** *seconds have passed since the last BISPDU was sent to the adjacent BIS, and start with any sequence number. The choice of the initial value of the sequence number is a local matter.*

c. In item ″a″, change ″established″ to ″establishes″.

9. Clause 7.6.1b and 7.6.1.2f, page 27 (Minor Technical): The contents of the header fields in the OPEN PDU are underspecified, so clarification is needed.

a. Change the 2nd sentence of 7.6.1.1b to read:

> The sequence field of the OPEN PDU shall contain the Initial Sequence Number (ISN); the Acknowledgement and Credit Available fields shall contain the value 0; and the Credit Offered field shall contain the initial flow control credit.

b. Add a new second sentence to 7.6.1.2 f, as follows:

> The value of the Credit Available field shall be set according to the procedures of clause 7.6.3, item b.

10. Clause 7.6.1.2, page 27 (Minor Technical): The FSMs in the DIS text allow the receipt of a KEEPALIVE, RIB REFRESH, or UPDATE PDU to trigger the transition from the OPEN-RCVD state to the ESTABLISHED state. However, they only call for the transmission of a KEEPALIVE PDU in response to a received OPEN (while in OPEN-RCVD or OPEN-SENT) with a KEEPALIVE. The ability to respond with an UPDATE or RIB REFRESH PDU can speed up the process of opening the connection by eliminating a non-information bearing exchange of KEEPALIVE PDUs. Therefore, we suggest the following changeg:

a. Change the first sentence of 7.6.1.2f to read ″...shall send a KEEPALIVE, RIB REFRESH, or UPDATE PDU that acknowledges...″

b. Change Table 2 for reception of an OPEN PDU with no errors while in OPEN-RCVD or OPEN-SENT states to read as follows: If ACK is correct, S=ESTABLSIHED, A=Send KEEPALIVE, UPDATE, or RIB REFRESH PDU″

11. Clause 7.6.1.3, page 29 (Minor Technical): In DIS 10747, the Hold Timer is used for two different purposes:

- When a BIS-BIS connection is being opened, the Hold Timer sets the amount of time that will be allowed for a BIS to receive a valid reply from the intended peer BIS
- After a connection has been established, the Hold Timer sets the amount of time that a BIS may remain in the ESTABLISHED state without receipt of a KEEPALIVE, UPDATE, or RIB REFRESH PDU from its peer BIS.

For certain deployment environments, the suitable time periods of each of these functions amy be vastly different: that is, a BIS may want a relatively long connection setup period, but also may wish to send KEEPALIVEs realtively frequently after the connection has been established.

We suggest that the Hold Timer should be used only in the ESTABLISHED state. Timeouts in the OPEN-RCVD state should be handled via a bounded number of retransmissions of the OPEN PDU.

Therefore, we suggest the following chagnes:

a. Strike 7.6.1.3c, and renumber accordingly.

b. Add a new item at the end of the list:

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

   **NOTE __: (Reproduce the text of existing Note 14 of the DIS text).**

c. Change Table 2 to reflect the changes above:

   - Hold Timer Expiry, OPEN-RCVD: S=OPEN-RCVD, A=none
   - Hold Timer Expiry, OPEN-SENT: S=OPEN-SENT, A=none

12. Clause 7.12.3.3, page 38 (minor technical): The checks made in the CD text to insure that the nesting order of confederations as depicted in the RD_PATH attribute are consistent with the information in managed object **RDC-Config** do not appear in the DIS text. To rectify this situation, insert a new third paragraph into item "b" of 7.12.3.3:

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

13. Clause 7.11, page 35 (minor technical): The handling of optional attributes needs to be expressed in a clearer fashion. To accomplish this, we recommend the following changes:

    - Reword the first sentence: *An UPDATE PDU that carries an NLRI field also carries a set of path attributes.*
    - Clause 7.11.1, first paragraph, item "c": Strike the word "even" in the last sentence of this item
    - Clause 7.11.1, second paragraph, item "a": Replace the existing text with the following:

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

    - Clause 7.11.1, second paragraph, item "b": Replace the existing text with the following:

      *If a route with a recognized optional transitive attribute is received and the route is to be propagated to other BISs, the optional transitive attribute may or may not be propagated with the route, according to the definition of the attribute. If the attribute is prop-*

*agated, then the local BIS shall not modify the value of the PARTIAL bit in the Flag field of the attribute.*

- Clause 7.11.1, second paragraph, item ″d″: Replace the existing text with the following:

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

14. Clause 7.16, Page 47 (minor technical):

    For clarity and completeness, the description of the Decision Process should note that potential loop-forming routes should not be used as input to the Decision Process. To accomplish this aim, we suggest adding the following text after the second paragraph of 7.16 (ending with ″with the highest degree of preference.″):

    *Routes that could form routeing loops must be ignored by the Decision Process. Therefore, any route that was a) received from a BIS located in an adjacent routeing domain and b) contains in its RD_PATH attribute a path segment of type RD_SEQ or RD_SET that contains the RDI of the local routeing domain or any RDC of which the local RD is a member is unfeasible, and shall be discarded by the Decision Process.*

    This change makes Note 26 superfluous, so it should be deleted.

15. Clause 7.16.3.1, page 50 (Minor technical): The restrictions presented in the last paragraph overly constrain the handling of overlapping routes. In particular, it outlaws the installation of a less specific route from a given neighbor if the more specific route from that *same neighbor* is not also installed. However, as long as a more specific route is installed by the local BIS, even if from a *different neighbor*, then no NPDUs whose destination addresses lie in the overlapping region (that is, destinations listed in the *uninstalled* more specific route of the given neighbor) will be forwarded to the given neighbor: the ″longest match″ rule will insure that such NPDUs are forwarded to the ″different neighbor″, whose more specific route has been installed.

    Therefore, we suggest that the text in the last paragraph of 7.16.3.1, including the list, be replaced with the following new text:

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

    *Therefore, when presented with overlapping routes from a given neighbor BIS (A), the local BIS has the following options to insure that its Adj-RIBs-Out and its FIBs do not alter the semantics of the routes advertised by the neighbor BIS (A):*

    1. *Install both the less specific and more specific routes received from the given neighbor (A)*
    2. *Install the more specific route received from the given neighbor (A) and reject A′s less specific route*
    3. *Install the non-overlapping part of the less specific and more specific routes received from the given neighbor (A)*

*4. Install a route formed by the aggregation of the less specific and the more specific route received from the given neighbor (A)*
*5. Install the less specific route received from the given neighbor (A), and also install another route received from a different neighbor (B) that is simultaneously:*
   - *more specific than A's less specific route, and*
   - *less specific than A's more specific route.*

*6. Install neither of the routes received from A.*

16. Clause 7.21.2, page 57, item ″c″ (Minor Technical): To aid in problem diagnosis, it would be helpful to include the offending RDI in the IDRP ERROR PDU that reports ″Bad_Peer_RD″. We suggest inserting the following as a new second sentence in item ″c″:

    *The value of the erroneous RDI is returned in the Data field of the IDRP ERROR PDU, encoded as a <length, RDI> pair. ″Length″ is a one octet field containing a positive integer that gives the number of octets used for the following ″RDI″ field.*

17. Clause 7.21.3, page 7, item ″g″ (Minor Technical): To aid in problem diagnosis, it would be helpful to report the offending RDI in the IDRP ERROR PDU. We suggest replacing the next-to-last sentence of item ″g″ with the following:

    *The data field of the IDRP ERROR PDU shall report the first RDI that indicated a loop. This RDI shall be followed immediately by the complete RD_PATH attribute. The encoding shall be: length, RDI, Offending RD_PATH attribute>, where:*

    - *″length″ is a one octet field that gives the length of the in octets of the immediately following RDI field*
    - *″RDI″ is the RDI that was detected as creating the loop*
    - *RD_PATH is the octet string that encoded the value field of the offending RD_PATH attribute in the received UPDATE PDU (see clause 6.3)*

18. Clause 7.21.3, page 58 (Minor Technical): There is no error checking for the occurrence of multiple instances of a given path attribute within a single UPDATE PDU. We suggest adding a new item to the list, as follows:

    *k) If an UPDATE PDU contains more than a single instance of any path attribute, then the error subcode shall be set to Malformed_Attribute_List. No further processing shall be done, and all information in the UPDATE PDU shall be discarded.*

    There is no checking for the occurrence of duplicated path attributes. We suggest adding a new item to the list:

    *If an UPDATE PDU contains more than one instance of a path attribute of a given type, the BIS shall send an IDRP ERROR PDU with error subcode DUPLICATED_ATTRIBUTES. The data field of the IDRP ERROR PDU shall list the type codes of all such duplicated attributes.*

    There is no error checking for illegal RD_PATH segment types. We suggest adding a new item to the list:

    *If the RD_PATH attribute contains an illegal segment type, the BIS shall send an IDRP ERROR PDU, with error subcode ILLEGAL_RD_PATH_SEGMENT. The data field of the IDRP ERROR PDU shall reproduce the encoding of the offending segment of the RD_PATH attribute, as it appeared in the received UPDATE PDU.*

19. Clause 8.4, page 60 (Minor Technical): The description of encapsulation in item ″b1″ of this clause fails to mention what should be done with the following parameters of an ISO 8473 NPDU, when

present: segmentation permitted, error report flag, and lifetime. We suggest that the second sentence (″The QOS parameter of the encapsulating...″) be replaced with the following new text:

> *Copy the following, when present in the header of the encapsulated (inner) NPDU, to the header of the encapsulating (outer) NPDU: QOS Maintenance parameter, Segmentation Permitted Flag, Error Report Flag, and PDU Lifetime field. When the inner NPDU is decapsulated, replace its PDU Lifetime field with PDU Lifetime field of the outer NPDU.*

20. Clause 11.3, page 66 (Minor Technical): Since the OPEN PDU from an adjacent BIS includes information on the maximum-sized BISPDU that it will send, there should be a corresponding managed object listed in clause 11.3, and an attribute description should be included in clause 11.4.

21. Clause 11.9, page 78, item ″Ribattributes″ (Minor Technical): The ASN.1 description for ″RIBattributes″ in 11.9 does not reflect the characteristics of a valid RIB-Att, as defined in 7.11.2: in particilar, the ASN.1 notation does not show that a valid RIB-Att can consist of at most three distinguishing attributes.

    To rectify this situation, the following changes should be made to 11.9:

    a. Replace the definition of ″Ribattributes″ on page 78 with:

    *Ribattributes ::= SEQUENCE {*
    *priority [0] EXPLICIT Priority OPTIONAL,*
    *security [1] EXPLICIT SEC OPTIONAL,*
    *qosmaint [2] EXPLICIT QOS OPTIONAL }*

    b. Add the following new items to 11.9:

    *SEC ::= CHOICE { ssSEC[0] EXPLICIT Ribattsec,*
    *dsSEC[1] EXPLICIT Ribattsec }*

    *QOS ::= CHOICE { global[0] EXPLICIT GLOBAL,*
    *ssQOS[1] EXPLICIT QOSTV,*
    *dsQOS[2] EXPLICIT QOSTV }*

    *GLOBAL ::= ENUMERATED ( delay(0), expense(1), capacity(3), error(4) }*

    *QOSTV ::= SEQUENCE { preflgth NSAPorefixLength,*
    *prefix NSAPpreifx,*
    *qOSlgth QOSlength,*
    *qOSval QOSvalue }*

22. Clause 12.2.2, Page 80 (Minor Technical): The clause reference given in ″Supporting RDCs″ points to a non-normative clause. Furthermore, upon review, the USA discovered that all elements necessary to support confederations are already mandatory elsewhere in the standard: for example, every BIS must support managed object **RDC_Config**, and every BIS must construct RD_PATHs based upon the entered/exited confederations (see 7.12.3ff). Hence, we conclude that support for confederations is already a mandatory feature of the protocol.

    Therefore, clause 12.2.2 should be deleted in its entirety.

23. Annex H, Page 94 (Minor Technical): This informative annex is no longer correct with respect to the DIS-level text. Originally, it illustrated very simple syntax, which was appropriate for the early working draft text of IDRP. However, in view of the changes to the base text during its progression, its usefulness is now very limited—in fact, its simplicity could be a source of confusion. Therefore,

a replacement annex, contained in Appendix A, “A New Policy Syntax Annex for IDRP” on page 23 of this ballot comment, is offered as a replacement.

## ***Major Editorial Comments***

24. Clarification of the Processing of Received BISPDUs (Major Editorial): The current text describing the processing of received UPDATE PDUs is organized in a somewhat confusing fashion. For example, the text in the description of the flow control process (7.5.3) says to pass in-sequence packets to the Receive Process, but the text for the receive process itself (7.20) describes overall flow without explicitly calling out which types of BIS PDUs are subject to sequencing constraints. Also, the text for the Update-Receive Process (7.14) describes the processing of other BISPDU types in addition to just the UPDATE PDUs.

    The USA suggests that this material could be presented in a more coherent fashion if the following changes were made. Note that these changes, although numerous, are all editorial in nature, and do not change the operational characteristics of the protocol. Finally, all clause references refer to the numbering as it is in DIS 10747, not to the numbering that will apply after the suggested rearrangements are carried out.

    Suggested changes:

    a. Move clauses 7.20 and 7.6 so that they precede clause 7.5: that is, the new order will be 7.4(existing), 7.20(moved), 7.5(moved), 7.5(existing). Renumber clauses and cross-references as required.

    b. Clause 7.20: replace the second sentence of the third paragraph (″This BISPDU shall be passed...″) with the following new text: *This BISPDU shall be passed to the IDRP Finite State Machine described in 7.6.1.*

    c. Add a new paragraph after the first paragraph of 7.6.1:

        *BISPDUs passed to this finite state machine are subject the flow control procedures of 7.5.3 if the FSM is in the ESTABLISHED state. When the FSM is in the ESTABLISHED state, only BISPDUs that are not discarded by the flow control process are processed by the FSM. In all other states, all BISPDUs are processed directly by the finite state machine without being subject to flow control procedures.*

    d. In clause 7.14, replace the first paragraph and its items ″a″, ″b″, ″c″, and the first sentence of ″d″ with the following new text:

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

        *For a feasible route, the Adj-RIB-In ...″*

       Note that this promotes the contents of ″d″ to main text, and thus the remaining list items need to renumbered accordingly.

    e. In 7.5.3, in the first, third, and fourth paragraphs, change the words ″passed to the Receive Process″ to ″passed to the Finite State Machine described in 7.6.1″

***Minor Editorial Comments***

25. Clause 6.2, page 12 (Minor Editorial): The table that shows the structure of the RIB-AttsSet uses language such as ″Number of Distinguish Attributes in First Set″. For clarity, we suggest changing ″Set″ to ″RIB-Att″ throughout this table.

    The order in which the distinguishing attributes of a given RIB-Att are listed is immaterial (because the RIB-Att is defined as a set, not as a sequence). For clarity, we suggest adding the following sentence to the last paragraph on the bottom right-hand column of page 12:

    > *Since a RIB-Att consists of a set of distinguishing attributes, there is no significance to the order in which the distinguishing attributes of a given RIB-Att are listed.*

26. Figure 5, page 14 (Minor Editorial): The NLRI information within the UPDATE PDU can carry both OSI and non-OSI addressing, but the figure was not updated to reflect this. A suggested replacement figure is shown in Figure 1 on page 20.

27. Clause 6.3, page 14 (Minor Editorial): Since the NLRI caters to carriage of non-OSI addresses, it would be useful to insert a note stating that it also caters to carrying locally defined path attributes as well--that is, note that some of the code space for the ″type″ field of the path attributes is not intended to be globally understood.

    Therefore, add a ″NOTE″ immediately after the last sentence on page 14, and number accordingly:

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

28. Clause 6.3, page 15 (Minor Editorial): Change ″are″ to ″is″ in the first sentence after item ″Value″.

    Insert the words *a well-known mandatory attribute* between ″The RD_PATH attribute is″ and ″composed...″ in the first sentence of item ″c″ (RD_PATH).

29. Clause 7.5.3a, page 25 (Minor Editorial): Since packet sequencing applies to all BISPDUs except the IDRP ERROR PDU, the last sentence should be amended as shown:

    > ...for an inbound *OPEN, UPDATE, KEEPALIVE, CEASE, or RIB REFRESH PDU* received from the peer BIS;...

30. Clause 7.5.3b, page 25 (Minor Editorial): The language in the third sentence of item ″b″ (″If an UPDATE or RIB REFRESH...incremented by one.″) is not clear on when the incrementing should actually take place To be consistent with the handling of KEEPALIVE PDUs, as described elsewhere in this clause, the sequence number should be incremented after the UPDATE or RIB REFRESH PDU is generated, but before it is actually sent to the peer BIS and before any other BISPDU is generated.

    Hence, the third sentence should be amended as follows, for clarity:

    > *When an UPDATE or RIB REFRESH PDU is to be sent, the local BIS shall generate the contents of the BISPDU based on the current value of the lower window edge. The local BIS shall increment the local window edge by one before it transmits the BISPDU to the peer BIS and before it generates any other BISPDUs; when a BISPDU other than an UPDATE or RIB REFRESH PDU is to be sent, the lower window edge shall not be incremented.*

31. Clause 7.6.2, page 31 (Minor Editorial): Since the FSM shows that expiration of the Hold Timer can cause a connection to be closed, this should be noted in the text by adding the words ″ *by expiration of the Hold Timer,*″ immediately before the last comma in the first sentence.

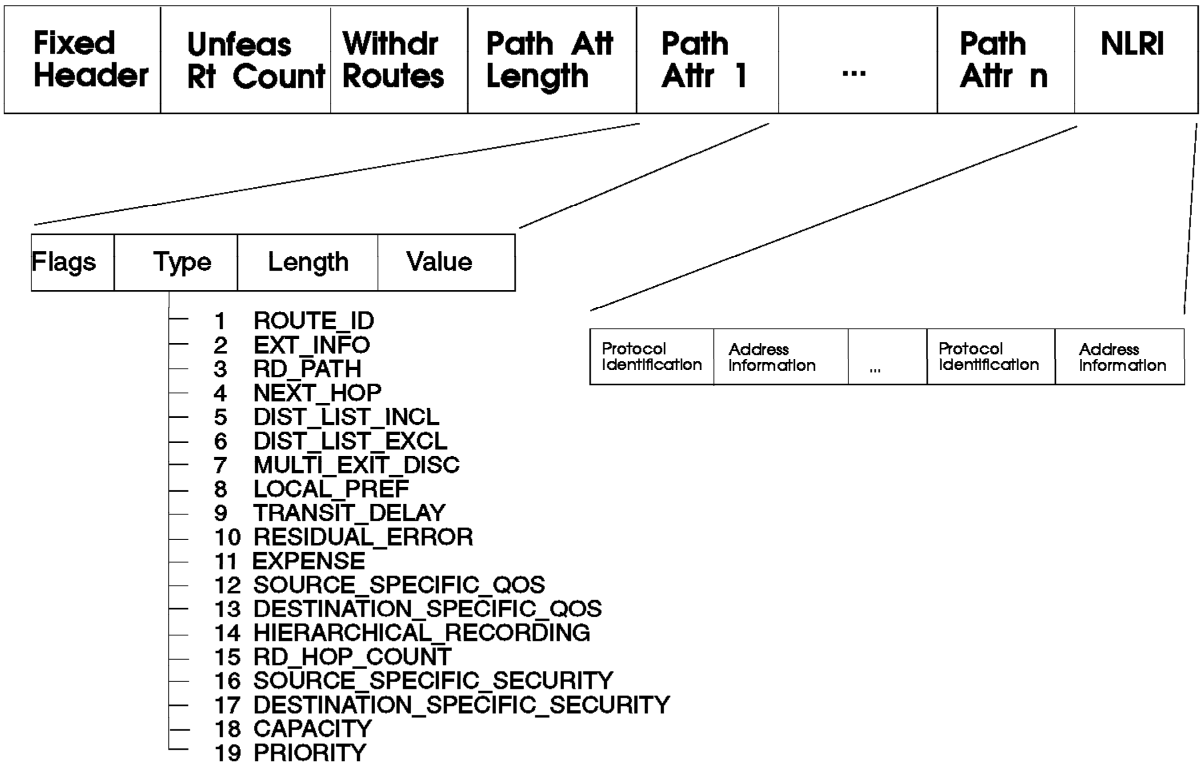


Figure 1. Replacement for Figure 5 of DIS 10747

In items ″a″ and ″c″, the phrase ″deallocate all resources...″ is used, but not defined anywhere Furthermore, the normative requirement is only that the FSM enter the CLOSE-WAIT state. Hence, the phrase ″shall deallocate...peer BIS″ in item ″a″ and the phrase ″shall deallocate...associated with it″ in item ″c″ should be deleted.

For clarity and consistency with clause 5.6, item ″c″ on page 8, it would also be helpful to insert a new final paragraph in 7.6.2:

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

Similarly, the words ″shall allocate a connection record″ should be deleted from 7.6.1.1b, since a connection record is not defined within IDRP.

32. Clause 7.10.1, page 33 (Minor Editorial): The important point about the RIB-AttSets supported by BISs in the same routeing domain is that they all list the same RIB-Atts, but there is no requirement that they be listed in exactly the same order. To clarify this, we suggest changing the words ″shall be identical...″ in the first sentence of the third paragraph to ″shall contain the same RIB-Atts...″.

33. Clause 7.13, page 46 (Minor Editorial): Insert the word ″to″ immediately after the word ″respect″ in the 4th line of the second paragraph.

34. Clause 7.14, page 47, item ″d)2)ii″, (Minor Editorial): For clarity, it needs to be pointed out that items ″ii″ and ″iii″ differ in that ″iii″ assumes identical path attributes, while item ″ii″ assumes that some of the non-distinguishing path attributes are different between the new route and the earlier route. Hence, add the italicized words immediately after ″...contained in the Adj-RIB-In″:

    ...contained in the Adj-RIB-In *and the non-distinguishing path attributes of the new route differ from those of the earlier route,...*

35. Clause 7.14, page 47, item ″d)2)v″, (Minor Editorial): Item ″v″ does not explicitly mention what to do with the earlier (more specific) route, relying on the absence of text to indicate that there are no normative requirements to take any action with respect to the earlier route. For clarity, it may be worthwhile to append an informative sentence: ″The earlier, more specific route remains unaffected.″

36. Clause 7.15.1, page 47 (Minor Editorial): The term ″local BIS″ in the second sentence of the first paragraph is imprecise: it actually refers to the BIS that is advertising the route in question. Hence, we suggest replacing this sentence with the following:

    *The value of LOCAL_PREF for a particular route is generated by the BIS that advertises the route, and is equal to the degree of preference for that route.*

37. Clause 7.18.2.1, page 53 (Minor Editorial): For precision, the words ″are the same″ at the end of the first sentence should be replaced with ″are equivalent, as defined in clause 7.11.3.″

38. Clause 7.20, page 56 (Minor Editorial): In the last sentence of the clause, change the reference from ″clause 8.4, item b2″ to ″clause 8.4, item b1″.

39. Clause 7.21.1, page 56 (Minor Editorial): This clause doesn′t mention the minimum lengths of the CEASE or IDRP ERROR PDUs. The list should be expanded to note that the minimum lengths are 30 for a CEASE PDU, and 32 for the IDRP ERROR PDU.

40. Clause 7.21.3, item ″n″, page 58 (minor editorial): There an unresolved cross reference in 7.21.3. The reference should be to 7.12.3.3, item b.

41. Clauses 11.1 through 11.8, pages 63-76 (Minor Editorial): Throughout these sections, the objects identifiers defined in 11.9 (idrpoi, sseoi, moi, poi, proi, nboi, atoi, agoi, acoi, and noi) appear in many cases without the qualifier ″IDRP″ to disambiguate them. In all cases where this occurs in the ″REGISTERED AS″ constructions, make the following changes where necessary:

    - idrpoi --> IDRP.idrpoi
    - sseoi --> IDRP.sseoi
    - moi --> IDRP.moi
    - poi --> IDRP.poi
    - nboi --> IDRP.nboi
    - atoi --> IDRP.atoi
    - agoi --> IDRP.agoi
    - acoi --> IDRP.acoi
    - noi --> IDRP.noi

    No change is needed in those cases where the ″IDRP″ qualifier is already present.

42. Clause 11.9, page 76 (Minor Editorial): Change the object identifier ″aoi″ to ″atoi″ in order to be consistent with the usage in clause 11.4.

43. Clause 11.9, page 77 (Minor Editorial): The decimal point and the final digit for the integer value of NonWrappingCounter are incorrect. The correct value is: 18446744073709551615

44. Clause 11.9, page 78 (Minor Editorial): Change ″ICIT″ to ″IMPLICIT″ in the description of ″sourcespecificqos″ within ″RIBattvalue″.

45. Annex I, page 98 (Minor Editorial): In the first dashed list item, change ″con federation″ to ″confederation″.

46. Annex K, Page 103 (Minor Editorial): This annex is informative, and is no longer correct with respect to the DIS-level text. It has not been updated nor commented upon since the earliest working draft in which it appeared (SC6 N6387, November 1990). In its present form, it is more confusing than helpful. Therefore, as this annex is only informative, it is recommended that it be deleted in its entirety.

47. Annex J, pages 100-102, (Minor Editorial): Change the words ″that destined″ to ″that is destined″ in three places: First and third line of the last paragraph of the first ″dashed″ bullet item at the bottom of page 100, and the first line in the left-hand column of page 102.

# Appendix A. A New Policy Syntax Annex for IDRP

This annex describes an example of a policy syntax and its associated semantics for the protocol defined in this international standard. The example is intended to be informative: that is, alternative syntaxes with equivalent richness of functionality are not precluded, and other mechanisms may be needed to provide a fully functional configuration language.

## A.1 Overview

The policy information base allows routing domain administrators to control routing information usage and flow according to the policies of the domain. The policy information base is made up of three component sections, corresponding to three primary types of policy concerns that have been identified:

1. *Route preference* assigns a preference value to incoming routes; this is the "local selection policy" regarding routes. These policies determine which routes in the Adj-RIBs-In are selected for the LOC-RIB.
2. *Route aggregation* chooses routes for aggregation and expresses some control over how aggregation is performed. These policies select routes in the LOC-RIB that are to be advertised as an aggregate. These policies can affect routes sent to BISs internal and external to the domain.
3. *Route distribution* modifies and selects routes for redistribution; this expresses the domain's "transit policy". These policies control traffic through the domain by restricting which routes from the LOC-RIB are placed in the RIB-OUTs. Modifications may affect routes sent to internal or external BISs, however, selection policy only affects the distribution of routes to BISs external to the domain; internal BIS neighbors receive route information from the local BIS regardless of policy.

Each policy subsection is comprised of a list of *policy statements* that express the domain's policy. Although the policy statements of each section are different, all include a *route pattern* (which is a template for matching route attributes) and the associated *actions*. A domain administrator can use these "match + action" pairs to express the administrative policy of the routing domain.

### A.1.1 Preference Statement

The preference statement is identified by the "PREF" keyword, and has the following format:

```
PREF <route pattern template> [<local_cond>] [<bis>]
  = <preference value expression>
```

A PREF statement assigns a value to any route (from a BIS neighbor in an external domain) that matches the specified pattern. The assigned value determines the degree of preference that will be used in the Decision Process. This value is also used to generate the LOC_PREF attribute. Note that it is possible for the assigned value to be less than zero or greater than 255. Conversion from the assigned value to an eight-bit LOC_PREF attribute is a local matter. Routes received from internal BIS neighbors will already have a LOC_PREF attribute. The use of the LOC_PREF attribute as a basis for selecting the most preferred route is described in clause 17.12.8.

The components of a <route pattern template> are:

- <nlri>
- [<info_src>]
- [<path>]
- [<dist_att>]
- [<att_cond>&rbrk, where:.

**<nlri> :** Reachable destinations; matches if the actual route′s NLRI is a subset of the destinations specified by this template. The <nlri> must be present in the route pattern of every policy statement.

**<info_src> :** Can be ″idrp″|″ext″|″info_any″, which is matched base d on the presence/absence of the EXT_INFO attribute in a route. These tokens are optional; if not present, the default match is ″idrp″.

**<path> :** Regular expression over RDIs to match against the content of the RD_PATH attribute. A <path> is optional; if not present, the default matches any RD_PATH attribute.

**<dist_att> :** Specifies a set of distinguished attributes for a route match. The <dist_att> is optional; if not present, the pattern matches routes with any set of distinguished attributes.

**<att_cond> :** Provides matching/control for all other attributes, i.e. other than what is carried in RD_PATH, EXT_INFO path attribute, and the presence of distinguished attributes. This specifies conditions of route attributes that must be met for a route to match, e.g. (EXPENSE() < 10) && (! present(DIST_INCL)) might be the condition if the intent is to match a low-cost route which does not have certain re-distribution restrictions. No <att_cond> need be specified; if not present, the route pattern matches routes with any attributes.

Note that the route pattern template is found in all three types of policy statement (preference, aggregation, and distribution). A slightly different form is used in the aggregation policy statement, which is discussed below.

The PREF statement (actually, all policy statements) may also include ″local condition tests″, which allow policy to be sensitive to criteria not related to a route′s attributes (e.g. time of day). A <local_cond> is optional; if not present, routes are matched under any local conditions.

The specification of <bis> allows routes from different BIS neighbors to be assigned preferences differently. Any number of external BIS neighbors may be specified, and only routes received from these neighbors will be assigned a preference value by the statement.

The <preference value expression> is an integer arithmetic expression with operators ′+′, ′-′, ′*′, ′/′, and (similar to the C language) a conditional operator ′?′. The basic operands are constants, or predefined functions which return values based on the attributes of a route, e.g. hopcount(), capacity(), weighted_list(EXCL,<table>). The condition expression for the condition operator includes the logic operators ″&&″ and ″||″, and may include (1) tests for the presence of an attribute, (2) comparisons of integer expressions including attribute values, and (3) local condition tests. A <pref value> is required in all preference statements.

The order of PREF statements in a configuration file is significant; the first <route pattern> that matches an incoming route will assign the preference value. The list of PREF statements can be thought of as filters, each acting on particular routes; a routing domain administrator can make effective use of this first-match functionality by listing more specific route patterns early and more general patterns later. Hence, the ″filters″ start at a fine degree of granularity to assign preference to routes of particular importance, while other routes are handled by increasingly general ″filters″.

If a route does not match any <route pattern>, it is dropped and not considered by the IDRP Decision Process. Note that there may be over-riding operational criteria that dictate that the non-matched routes can not be handled in this manner.

The concept of decreasingly specific filters is useful for all of the policy sections: preference, aggregation, and distribution. As described below, more flexible control of the processing sequence for aggregation and distribution statements is possible, and necessary to concisely express policy.

### A.1.2 Aggregation Statement

The aggregation statement is identified by the ″AGGR″ keyword, and has the following format:

```
AGGR <route pattern> <local_cond> =
 [<recipient BIS>] <aggr_nlri> [″DONE″|″CONT″]
```

The <route pattern> specification of the aggregation statement is slightly different than the PREF statement <route pattern>. The only difference is that the <nlri> template will consist of two NLRI specifications separated by the ″MUST″ token, i.e. <nlri> ″MUST″ <nlri>. The first <nlri> is used to match routes that can be aggregated, while the second <nlri> specifies NLRI which must be present for the aggregate route to be instantiated. Either of the <nlri> specifications may omitted, but not both. If the second <nlri> is omitted, the ″MUST″ token is not required.

The AGGR statement′s <local conditions> template has the same syntax and semantics as the PREF statement.

The <recipient BIS> of the aggregation statement indicates which external BIS′s RIB-OUTs are to receive the results of the statement′s route aggregation. One, several, or all external BISs may be specified to receive the aggregate route ″manufactured″ by an AGGR statement. In addition, an administrator can specify ″internal_bis″ to affect aggregation to all other BISs internal to the routing domain.

If a BIS is included in the recipient list, it will receive the aggregated route but not the component routes; if an aggregate is not instantiated to a particular BIS, it will receive all of the component routes. Note that by using additional AGGR statements (with more specific route matching templates), particular component routes may be advertised separately from the aggregate route. If the <recipient BIS> list is not specified, the default action is to announce the aggregate route to all external neighbor BISs; the default action will announce component routes to internal BISs.

The <aggr_nlri> specifies how the BIS determines which NLRI to advertise for the aggregate route. The two primary specifications are manual (″man″) or automatic (″auto″), with two additional tokens (″auto_short″ and ″auto_subset″) to specify variations of ″auto″; ″auto″ includes both of these variations. Automatically aggregated NLRI will only reduce routes if there is no loss of reachability information, i.e. it will only advertise a more general NLRI if it can algorithmically determine that the aggregate is not advertising NLRI other than those of the component routes. Domain administrators can also ″manually″ override the automatic aggregation and specify that aggregated route NLRI may include destinations not included in any component of the aggregate route. The ″manual″ option is primarily intended for use when additional (complete) information is known about the NLRI (e.g. when it is part of the address space under control of the routing domain). It is assumed such information is obtained by means outside of IDRP. For instance, using ″manual″ NLRI configuration, a domain that acts as an address assignment authority may announce a single prefix for all routes containing longer extensions of this prefix, even though portions of the address space may be unassigned, with no route available to some destinations advertised by the NLRI. Manually aggregated NLRI is determined by taking the longest common prefix of the set of NLRI specified by the route pattern <nlri>. Using automatic aggregation, the aggregate NLRI is computed to be the shortest NLRI prefix necessary to announce the component route′s NLRI (the aggregate NLRI is also the longest common prefix of the component routes). The two variations of ″auto″ are as follows: (1) ″auto_short″ will collapse several longer NLRI prefixes into a single common prefix based on the binary representation, e.g. XX:YY:0xF601:* - XX:YY:0xF60F:* will be advertised as XX:Y and (2) ″auto_subset″ will permit longer prefixes to be aggregated with shorter ones, e.g. XX:YY:ZZ:* would be aggregated with XX:YY:* into XX:YY:*.

Like PREF statements, the AGGR statements are applied in sequence (they are applied to the set of routes in the LOC-RIB). ″DONE″ and ″CONT″ provide control over additional processing of routes by subsequent AGGR statements. ″CONT″ is used to indicate that the aggregate route may be treated as a component route by later AGGR statements, and thus may be matched and further aggregated. ″DONE″ indicates that the aggregate is to be advertised as-is, and will not be considered as a component route for further aggregation. Specification of ″DONE″ or ″CONT″ is optional; the default case is ″DONE″.

[Note: Aggregated routes will have a preference value assigned by the policy PREF statements; just as incoming routes from other BISs, aggregated routes are processed by the route preference statements. If an aggregate route does not match a PREF statement template, no value is assigned and the aggregate is not instantiated.]

### A.1.3 Distribution Statement

The distribution statement is identified by the ″DIST″ keyword, and has the following format:

```
DIST <route pattern> [<local_cond>] = [<recipient BIS>]
  <select_action> [<modifications>] [″DONE″|″CONT]″]
```

The <route pattern> for the DIST statement is the same as the PREF statement <route pattern>, and <local_cond> serves the same function for the DIST statement as it does for the AGGR and PREF statements.

Similar to the AGGR statement, the <recipient BIS> specifies which RIB-OUTs are effected by the statement. The RIB-OUTs associated with the neighbors specified in <recipient BIS> may be affected in three ways by a DIST statement: (1) the route may be modified in these RIB-OUTs, (2) the route may be placed in, or removed from, the RIB-OUTs, and (3) the route may be marked as ″DONE″, so that it remains unaffected by further DIST statements.

The <select_action> can be ″select_on″, ″select_off″, ″select_only″, or ″modify″; these control whether a route is distributed to an adjacent BIS. If a route is selected for advertisement to a particular BIS neighbor, it will be placed in the associated RIB-OUT. By default, routes are not selected for advertisement until selected by a DIST statement. The semantics of the <select_action> effect this distribution as follows:

″**select_on**″ The route should be placed in RIB-OUTs associated with all specified neighbors, unless ″selected off″ by later DIST statement.

″**select_off**″ The route should not be placed in RIB-OUTs associated with the specified neighbors, unless ″selected on″ by a later DIST statement.

″**select_only**″ The route should be placed in RIB-OUTs associated with all specified neighbors, unless ″selected off″ by later DIST stmt; in addition, the route should not be placed in RIB-OUTs associated with BISs not in <recipient BIS>, unless ″selected on″ by a later DIST statement.

″**modify**″ Modify only; the selection status of routes are not effected by this DIST statement. Presumably, some routes matching this statement will also match, and be selected for distribution by, other DIST statements.

The effects of a <select_action> is applied only when <recipient BIS> indicates a BIS in an adjacent domain. It has no effect on distribution to BISs within the same domain as the local system.

Note that in most cases, only the routes in RIB-OUTs specified by <recipient BIS> will be affected by a DIST statement, however, there is one exception. The ″select_only″ action also indicates routes are not to appear in the RIB-OUTs associated with BISs not in <recipient BIS> list, and that these routes may or may not be considered for further DIST statement processing (in the excluded BISs) based on the DIST statement′s DONE/CONT token. Using ″select_only″ along with ″DONE″ allows one to concisely specify that only certain BISs are to receive particular routes, and as an additional effect, make certain these routes are not inadvertently selected for other BISs by a subsequent DIST statement that matches a more general route pattern.

A list of <modifications> statements indicates policy-driven changes to route attributes (e.g. DIST_LISTs, HIERARCHICAL RECORDING changes, etc). No <modifications> need be present; the default leaves routes unchanged.

″CONT″ and ″DONE″ have similar function as in the aggregation statement; they control whether routes matching a particular DIST statement may be affected by later DIST statements. ″CONT″ indicates that a matched route in a specified RIB-OUT is eligible for further modifications, ″DONE″ indicates no further DIST statement processing. Specification of ″DONE″ or ″CONT″ is optional; the default case is ″DONE″.

## A.2 Policy Configuration Language BNF

This section specifies the basic syntax for this example IDRP configuration language. This BNF tree does not include all terminal-symbol leaves; it is sufficient as an illustration of some minimal useful functionality, however, it is not complete.

The policy configuration language uses a ′#′ to denote a comment to the end of line. This convention is also used to provide comments throughout the BNF specification. This BNF uses square brackets, ′[′ and ′]′, as a notational convenience to indicate optional (zero or one occurrence) syntactic symbols. This BNF also uses curly braces, ′{′ and ′}′ and a ′|′ to indicate a choice of symbols.

A discussion of the semantics of this language can be found in A.1.1, “Preference Statement” on page 23 above..

### A.2.1 PREF Statement BNF

```
<preference_section> ::= <p_stmt_list>
 <p_stmt_list> ::= <p_stmt> ′;′ <p_stmt_list> | <empty>
 <p_stmt> ::= ″PREF″ <nlri> <route_pattern> <local_cond> [<bis>]
```

All of the symbols used by the <p_stmt> are also used in other places, and are defined in -- Heading ′COMBNF′ unknown --.

### A.2.2 AGGR Statement BNF

```
<aggregation_section> ::= <a_stmt_list>
 <a_stmt_list> ::= <a_stmt> ′;′ <a_stmt_list> | <empty>
 <a_stmt> ::= ″AGGR″ <nlri_2> <route_pattern> <local_cond>
′=′ [<bis>] <aggr_nlri> <done_cont>

<nlri_2>::= ′{′ <dest_list> [ ″MUST″ <dest_list> ′}′ ]
<aggr_nlri>::= ″auto″ | ″auto_subset″ | ″auto_short″ | ″man″
```

### A.2.3 DIST Statement BNF

```
<distribution_section> ::= <d_stmt_list>
 <d_stmt_list> ::= <d_stmt> ′;′ <d_stmt_list> | <empty>
 <d_stmt> ::= ″DIST″ <nlri> <route_pattern> <local_cond>
       ′=′ [<bis>] <select_action> <mods> <done_cont>
<select_action> ::= ″select_on″ | ″select_off″ |
 ″select_only″ | ″modify″
```

Policy- defined changes to route attributes are distinct from attribute updates that occur due to basic ″operational″ processing (e.g. HOP_COUNT is updated without regard to policy).

```
<mods> ::= ′{′ <mod_list> ′}′ | <empty>
 <mod_list> ::= <mod_statement> ′;′ <mod_list> | <empty>
 <mod_statement> ::= <multi_exit_statement> |
    <dist_list_statement> |
    <hrecord_statement>  |
    <next_hop_statement>

<multi_exit_statement> ::= ″set_multi_exit″ ′(′ <value> ′)′
```

init_hr() - If HIERARCHICAL_RECORDING attribute is not already present in route, add attribute to route and initialize to one (1) to limit distribution within RDC.

```
<hrecord_statement> ::= ″init_hr″ ′(′ ′)′
```

Add RDIs to INCL or EXCL list.

```
<dist_list_statement> ::=
"allow_dist" '(' <rdis> ')' |
"prohibit_dist" '(' <rdis> ')'

<next_hop_statement> ::=
"set_next_hop" '(' <net> <snpa> ')'
```

### A.2.4 Common BNF Symbols

This section describes common syntax components used by all three types of policy statements.

#### A.2.4.1 Route Attribute Matching Template

Reachability:

```
<nlri> ::= '{' <dest_list> '}'
<dest_list> ::= <dest> ',' <dest_list> | <dest>
<dest> ::=    "nlri_any"          |
  ["not"] <nsap> ':' '*'  |    ## prefix match
  ["not"] <nsap>          |    ## exact <nsap> match
  <empty>
```

Route matching template

```
<route_pattern> ::= <info_src> <path> <dist_att> <attrib_cond>
<info_src> ::= "idrp" | "ext" | "info_any"
<path> ::= '/' <<regular-expression over RDIs>> '/'
```

Distinguished attributes

```
<dist_att> ::= <empty> |
"dist_att_none" |
"dist_att_any" |
'(' <qos> <security> <priority> ')'
(If <empty>, default is "dist_att_any".)

<qos> ::= "qos_any" | <qos_list>
<qos_list> ::= <one_qos> <qos_list> | <empty>
(If <empty>, default is "qos_any".)

<one_qos> ::= "qos_none" | "error" | "expense" | "delay" |
"capacity" | <src_qos> | <dst_qos>
<src_qos>    ::= "srcqos"  <nsap>  <qos_value>
<dst_qos>    ::= "dstqos"  <nsap>  <qos_value>
<qos_value> ::=  ## TO BE DEFINED ##
```

```
<security> ::= "security_any" | <sec_list> <sec_list> ::= <sec_list> <one_sec> | <empty>
```

(If <empty>, default is ″security_any″.)

```
<one_sec> :: = ″security_none″ | <srcsec> | <dstsec>
<srcsec> ::= ″srcsec″ <nsap>
<dstsec> ::= ″dstsec″ <nsap>
```

Security-related BNF is subject to change as the protocol continues to develop.

```
<priority> ::= ″priority_any″ | ″priority″ | ″priority_none″| <empty>
```

The ″priority_any″ token matches routes in either case, whether; thef priority attribute is present, or if it is not. If <empty>, default is ″priority_any″.

### A.2.4.2 BNF: Numerical Expressions

```
<value> ::=
 <integer>     |
 <att_value>    |
′(′<value> ′)′ |
 <value> <integer_op> <value> |
 <case_statement>
<case_statement> ::=  ′(′ <case_cond> ′?′ <value> ′:′<value> ′)′
<att_value> ::=
″hopcount″   ″()″              | ## rd_hopcount
″pathweight″ ′(′ <table> ′)′    | ## weighted path
″listlen″   ′(′ {″INCL″|″EXCL″} ′)′              |
″listweight″ ′(′ {″INCL|″EXCL″} ′,′ <table> ′)′     |
 <att_value_name> ″()″
```

Returns the value carried by the attribute specified by the <att_value_name>.

```
<att_value_name> ::= ″multi_exit″ | ″loc_pref″ | ″priority″
 | ″delay″ | ″expense″  | ″error″ | ″capacity″
 | ″hier_rec″
```

This example of the PIB BNF does not deal with the following attributes: SRC_QOS, DST_QOS, SRC_SECURITY, and DST_SECURITY.

### A.2.4.3 BNF: Conditional Specification

There are three related types of conditions:

1. <attrib_cond> used when doing a route match; only tests/examines attributes of a route
2. <local_cond> used in policy actions; tests ″other″ (TBD) criteria (e.g. time of day)
3. <case_cond> used in case statement; may test attribute or local criteria

```
<local_cond> ::=
 <A_cond_LOCAL>              |
′!′ <local_cond>              |
′(′ <local_cond> ′)′            |
 <local_cond> ″&&″ <local_cond>  |
```

```
<local_cond> "||" <local_cond>

 <A_cond_LOCAL>  ::=   ## TO BE DEFINED
```

This is currently a place holder reserved for future use; one potential example is time of day.

```
<attrib_cond> ::=
 <A_cond_ATTRIB>              |
 '!' <attrib_cond>            |
 '(' <attrib_cond> ')'        |
 <attrib_cond> "&&" <attrib_cond> |
 <attrib_cond> "||" <attrib_cond>

<A_cond_ATTRIB> ::= <att_value> <compare_op> <value>    |
 "present" '(' <attribute_name> ')'  |
 <other_att_test>

<case_cond> ::=   <A_cond_CASE>            |
 '!' <case_cond>             |
 '(' <case_cond> ')'         |
 <case_cond> "&&" <case_cond>  |
 <case_cond> "||" <case_cond>

<A_cond_CASE>  ::= <A_cond_ATTRIB> | <A_cond_LOCAL>

<attribute_name> ::= "src_qos" | "dst_qos" |
 "dst_sec" | "src_sec" |
 "dist_incl" | "dist_excl" |
 "ext_info"  | "next_hop"  |
  <att_value_name>

<other_att_test> ::=  <next_hop_test>
```

This PIB BNF only defines the next_hop_test; others may be defined.

```
<next_hop_test> ::= "next_hop" '(' <next_hop_list> ')'
<next_hop_list> ::= <next_hop_match> ',' <next_hop_list>   |
 <next_hop_match>
<next_hop_match> ::= "next_hop_any"          |
 ["not"] <net> ':' '*'      |
 ["not"] <net> [<snpa>]    |
 ["not"] <net> <one_snpa>   |
```

One can attempt to match NEXT_HOP attribute against "any", a set of BISs (NET prefix), a particular BIS and optionally specific interfaces. Also one can match routes against local interface over which route was received.

### A.2.4.4 Other Common BNF Symbols

<bis> ::= "bis_all" | '{' <bis_list> '}' <bis_list> ::= <bis_item> ',' <bis_list> | <bis_item>
<bis_item> ::= "rdi" <one_rdi> | "bis" <net> |
"internal_bis" | "external_bis"

One can specify all BIS neighbors in an adjacent RDIrdi, single out a particular bis by NET, specify all internal BIS neighbors, or all external BIS neighbors.

<done_cont> ::= "DONE" | "CONT" | <empty>
(Default <empty> is DONE)

<table> ::= '{' <table_list> <table_default> '}'
<table_list> ::= <table_pair> <table_list> | <empty>
<table_pair> ::= '(' <one_rdi> ',' <integer> ')'
<table_default> ::= '(' "default" ',' <integer> ')' | <empty> p.List of interfaces of this BIS;

<snpa> ::= '{' <snpa_list> '}'
<snpa_list> ::= <one_snpa> ',' <snpa_list> | <one_snpa>

Routing Domain Identifiers;

<rdis> ::= '{' <rdi_list> '}'
<rdi_list> ::= <rdi_list> ',' <one_rdi> | <one_rdi>

## A.3 Simple Example

This example is provided to make the intended use of the policy configuration language more clear. Note that this example is incomplete, and at best only marginally realistic; it is intended to illustrate the basics of the policy configuration statements for purposes of this overview.

Throughout this text we refer to the set of distinguished attributes which has no QOS attribute, no priority attribute, and no security attribute as the default set of distinguished attributes. This is the distinguished attribute set specified by "dist_att_none".

Given the following portion of an internet:

```
     \    /
      \  /
        1 -------- 2 ----- 4
  |     |          |           1,2,3,5,6,7: transit domains
  |     |          |                   4/8: stub domains
   6 --- 3 -------- 5 ----- 8
          |
          |
     --- 7
```

Assume that each routing domain has exactly one BIS that communicates with all adjacent domains′ BISs.

### A.3.1 Transit Domain 3

Example policies of transit domain RD #3 might be as follows:

1. RD #3 only accepts IDRP originated routes. It supports two sets of distinguished RIB_ATTs: the default set (no distinguished attributes) and the set having only the CAPACITY QOS attribute.
2. Routes with CAPACITY QOS must travel via RD#6; CAPACITY must be greater than 15.
3. For routes with no distinguished attributes, prefer routes through transit domain 1, however preference should be given to routes with short paths; large hop counts on a route via RD#1 may cause a shift to another transit domain.
4. CAPACITY QOS routes are only offered to some domains (RDs #5, #8), and are restricted from being propagated further (i.e. via the DIST_LIST_INCL attribute).
5. All routes with no distinguished attributes are re-distributed to every neighbor RD; hierarchical recording is desired to limit distribution of all of these routes (the specific RDC membership information is irrelevant for this example).
6. Any route (default or CAPACITY) which pass through transit RD#9 (not pictured) can only be redistributed to some domains (RD#2, RD#5, RD#8).
7. All routes carrying NLRI of the address space controlled by domain #3 (XX:YY:3:*) will be aggregated (regardles s whether or not aggregated routes include NLRI for all of this space). In addition, routes carrying NLRI for RD#5 NSAPs (XX:YY:3 :5:*) will be announced separately. All routes for default dist_atts will be aggregated algorithmically. A ″default route″ (zero length NSAP) will be advertised for CAPACITY QOS routes (although distribution of this route will be limited to particular domains, i.e. RD#5, by the select/modify policy section).

### A.3.2 Policy Configuration Example

The following is one example of an expression of the above policies using this configuration language. The next subsection, examines and discusses each line of this configuration example in detail.

This example assumes that the policy language is case insensitive.

```
PREF {nlri_any} / .* 6 / (CAPACITY security_none priority_none)

(CAPACITY() > 15) = 50;

PREF {nlri_any} / .* 1 / dist_att_none = 255 - hopcount();

PREF {nlri_any} /.*/     dist_att_none = 245 - hopcount();

AGGR {XX:YY:3:5:*} /.*/ dist_att_none  =  man;

AGGR {XX:YY:3:*}   /.*/ dist_att_none  =  man;

AGGR {nlri_any}    /.*/ dist_att_none  =  auto;

AGGR {nlri_any}  /.*/ (CAPACITY security_none priority_none) = man;

DIST {nlri_any}   /.* 9 .*/ =
```

```
modify {allow_dist({RD#2, RD#5, RD#8});} CONT;
DIST {nlri_any}  /.* 6/ (CAPACITY security_none priority_none)
CAPACITY() > 15) = {BIS#5} select_only {allow_dist(RD#5, RD#8);};
DIST {nlri_any} /.*/  = select_on {init_hr();};
```

## A.3.3 Discussion

Each policy statement given in section 4.2 is discussed. In most cases, the optional parts of the BNF have been omitted if the default action is appropriate to represent the example policy. For brevity, these defaults will be mentioned in the discussion only once, at the statement where they are first encountered.

### A.3.3.1 Preference Statement Discussion

Recall that the sequence of statements is significant for determining the application and processing of all types of policy statements. Preference statements are the least complex of the three; routes are simply assigned the preference associated with the first <route pattern> that is matched (the other statements′ processing and application sequence are discussed later in this text).

The first PREF statement matches routes to any destination, indicated by {nlri_any}. No token for information source is present, so by default only routes where the information source was IDRP are matched (i.e. routes where no EXT_INFO attribute is present). The third expression ″/ .* 6 /″ matches any RD_PATH attribute where the last ″hop″ was from RD#6.

```
PREF {nlri_any} / .* 6 / (CAPACITY security_none priority_none)
 (CAPACITY() > 15) = 50;
```

The 3-tuple (CAPACITY security_none priority_none) indicates a route is to match if it corresponds to the RIB_ATT which has CAPACITY QOS, no security, and no priority. Finally, the attribute conditions only allow routes with CAPACITY attribute greater than 15 to be matched. There are no local conditions to be considered, so nothing is specified and by default routes are matched under any local conditions. Note that none of the actions in this example are dependent on local conditions, so this will be ignored for the rest of the example. Routes matched by this pattern are simply assigned a preference of 50.

The second PREF statement also matches routes to any destination, if the routing information source was IDRP. The statement matches routes received directly from RD#1, by examining the route′s RD_PATH attribute. The ″dist_att_none″ is specified, so only routes which have no QOS attribute, no priority attribute, and no security attribute will be matched. There are no other attribute or local conditions to meet, which is the default if nothing is specified.

```
PREF {nlri_any} / .* 1 / dist_att_none = 255 - hopcount();
```

This statement assigns a route preference based on the HOP_COUNT value plus a constant. The constant (255) is relatively ″good″ (relative to 245 in the next statement) so that routes through RD#1 are preferred (per policy C above). Since routes will be assigned preference by the the first <route pattern> matched, a path through RD#1 matching this pattern will not have a value assigned by the next statement, even though it has a more general <route pattern> and also would be a correct match.

The next PREF statement matches any route with no distinguished attributes, again, only if the information source was IDRP.

```
PREF {nlri_any} /.*/     dist_att_none = 245 - hopcount();
```

Routes matching this pattern are assigned a preference based on the hop count and a relatively ″bad″ constant (245), so that these routes are preferred less than routes through RD#1 (which match a previous PREF statement).

Examining the last two preference statements, per policy C routes through RD#1 are preferred unless the path length (hop count) is worse (by ten hops or more).

### A.3.3.2 Aggregation Statement Discussion

Aggregation statements are also processed in the order that they appear, however, their processing and application is not simply based on first match. An AGGR statement may be marked with ″CONT″ to indicate that the aggregate route may act as a component for subsequent AGGR statements. Alternatively, ″DONE″ indicates that an aggregate should be installed/advertised, and not considered in further aggregation processing.

The first aggregation statement matches routes with a specific set of destinations, {XX:YY:3:5:*}, and announces the aggregate route with manually configured NLRI. The longest common NLRI prefix specified is XX:YY:3:5, so this will serve as the NLRI for the aggregate route. Using ″manual″ aggregation, this aggregate is instantiated whether or not the NLRI of the matched component routes include all destinations implied by the prefix XX:YY:3:5.

```
AGGR {XX:YY:3:5:*} /.*/ dist_att_none = man;
```

Any number of routes may match this pattern, and are replaced by a single aggregate route. No recipient <bis> are specified, so by default the aggregate route (rather than the components) is announced to all external BIS neighbors. The default action, ″DONE″, requires that the aggregate route be distributed without undergoing further aggregation. Hence, routes to these destination NSAPs are announced separately from the rest of the XX:YY:3:* NSAPs (which are aggregated below).

The second AGGR statement is almost identical to the first.

```
AGGR {XX:YY:3:*} /.*/ dist_att_none = man;
```

Routes to a specific set of NSAPs are matched and aggregated; these destinations are a superset of those matched by the previous AGGR statement. This construct (two AGGR statements with overlapping NLRI) can be used to make certain that particular longer prefixes are announced separately from a more general aggregate prefix.

The third aggregation statement matches routes to any destination, with any RD_PATH, with no distinguished attributes, and no additional attribute or local conditions.

```
AGGR {nlri_any} /.*/ dist_att_none = auto;
```

These routes are to be aggregated automatically; that is safely and algorithmically such that the aggregated NLRI does not include more NSAPs than the component routes did. By default, this statement affects the routes that are announced to all external BIS neighbors.

The fourth AGGR statement matches routes to any destination, with any RD_PATH, if they have distinguished attributes that include only CAPACITY QOS.

```
AGGR {nlri_any}  /.*/ (CAPACITY security_none priority_none) = man;
```

Manual aggregation will use the longest common prefix of the specified NLRI as the aggregate route's NLRI. This statement matches routes to any destination, so the aggregate NLRI is a "default" route (route with zero length NLRI). H4 id=distdis.Distribution Statement Discussion

Distribution statements are the most complex of the policy statements; they control both the selection and modification of routes for re-distribution. DIST statement processing is sequential, and like the AGGR statement, "CONT" and "DONE" affect the processing of a route by policy statements. If a DIST statement specifies "DONE", routes will not be affected by any subsequent DIST statements. A "CONT" token indicates that routes should be affected by the next DIST statement that is matched. Using the idea of increasingly or decreasingly specific <route pattern> templates in combination with "DONE" and "CONT" to selectively prohibit further processing of some routes, a wide range of policy requirements can be concisely expressed.

The first DIST statement from the example matches routes to any destination, originated by IDRP (the default match), where RD#9 is in the RD_PATH. These routes can have any distinguished attribute set (any distinguished attribute is the default match), and no additional attribute or local conditions need to be satisfied.

```
DIST {nlri_any}   /.* 9 .*/ =
 modify {allow_dist({RD#2, RD#5, RD#8});} CONT;
```

This statement does not indicate a <bis> list, so by default all external BIS neighbors' RIB-OUTs are affected by this statement. The "modify" indicates that route attributes are to be modified, but the route's "selected status" will remain unchanged by this DIST statement. One modification is performed which restricts the distribution of these routes (per policy F above) by altering the DIST_LISTs to only allow certain domains to receive this route. The statement indicates "CONT", so these routes may be further modified, and/or selected for distribution to adjacent BIS, by subsequent DIST statements.

The second DIST statement matches routes to any destination with distinguished attributes (CAPACITY security_none priority_none).

```
DIST {nlri_any}  /.* 6/ (CAPACITY security_none priority_none)
 (CAPACITY() > 15) = {BIS#5} select_only {allow_dist(RD#5, RD#8);};
```

The {BIS#5} indication along with "select_only" specifies that the matched routes are to be selected for distribution only to BIS#5; an additional effect of "select_only" is to explicitly mark these routes as NOT distributable to all other BISs (all but BIS#5). The default action, "DONE", will keep these routes from being affected by other DIST statements. The "DONE" combined with the "select_only" will also prevent these routes from being matched and possibly placed in the RIB-OUT for distribution to other BISs (i.e. other than BIS#5). Using the "allow_dist()" function, this statement modifies the DIST_LISTs of matched routes to restrict further redistribution to domains RD#5 and RD#8.

The third DIST statement matches routes to any destination, with any RD_PATH; these routes can have any distinguished attributes, and no additional attribute or local conditions need to be satisfied.

DIST {nlri_any} /.*/ = select_on {init_hr();};

The RIB-OUTs for all neighboring BISs are affected by this statement, which selects the matched routes for distribution (″select_on″) and modifies the hierarchical recording attribute so it is initialized to ″1″ (only if the attribute is not already present and thus can be initialized according to operational procedures).

### A.3.3.3 Operational Example

Consider a route with the following attributes that arrives at our BIS configured with the above Policy Information Base (PIB):

nlri(10:66:*) rd_path(6 22 10) hopcount(15)

It is a route with no distinguished attributes, which matches only one of the PREF statement route patterns:

PREF {nlri_any} /.*/ dist_att_none

and is assigned a preference of 230 (245-hopcount()) by the this PREF statement. Consider a second route to the same destination NLRI with attributes:

nlri(10:66:*) rd_path(1 44 9 16 10) hopcount(20)

It also has no distinguished attributes. It matches two PREF route patterns, however, only the first match is considered (first match).

PREF {nlri_any} idrp /.* 1/ dist_att_none

Because the route is through RD#1 it is a preferred route, and is assigned a preference of 235 (255-hopcount()). Both of these two routes are for the same set of destination NLRI; the second route, with preference value of 235 would be chosen over the route with preference value 230. If these were the only two routes to these destinations, the preferred route would be installed in our LOC_RIB and FIB.

Now aggregation policy must be considered to see how the route is to be announced. The preferred route that was placed in the LOC_RIB:

nlri(10:66:*) rd_path(1 44 9 16 10) hopcount(20)

matches one AGGR statement route pattern, which specifies automatic aggregation for those routes where it is possible:

AGGR {nlri_any} /.*/ dist_att_none = auto;

Depending on what other routes and aggregates are installed, this route may be announced individually, or it may be part of an already instantiated aggregate. For instance, if there is already an (aggregate) route to nlri(10:*), the example route could be included in the nlri(10:*) aggregate; the example route would be installed in the LOC-RIB and FIB (so packets are forwarded correctly), and then a new aggregate (made up of this route and the old aggregate) would be composed. If a new aggregate were to be generated, a new preference value would be assigned by the PREF policy statement processing. Whether this route is aggregated with other routes, or maintained individually, it must be selected by a DIST statement before it will be announced.

Assuming that there is no aggregate for this route, it is installed in the LOC-RIB and FIB as-is, and must be considered for redistribution. Again, the route:

```
nlri(10:66:*) rd_path(1 44 9 16 10) hopcount(20)
```

is matched against route patterns. It matches the first DIST statement:

```
DIST {nlri_any} /.* 9 .*/ =
 modify {allow_dist({RD#2, RD#5, RD#8});} CONT;
```

which requires the route be modified before it is re-distributed. Applying the modifications the route becomes:

```
nlri(10:*) rd_path(1 44 9 16 10) dist_list_incl(2,5,8) hopcount(20)
```

Since this DIST statement does not terminate distribution processing, (″CONT″), other DIST statements may be matched. At this point the route has been modified, but has not been selected for distribution (″modify″ rather than ″select_on″ or ″select_only″ was specified). The route also matches the following DIST statement:

```
DIST {nlri_any} /.*/ = select_on {init_hr();};
```

which modifies the route (initializing the HIERARCHICAL_RECORDING attribute since it′s not already set), and selects the route for distribution (to all external BIS neighbors). This DIST statement (by default) specifies ″DONE″, so no further distribution processing is applied to this route. Note that other changes to the route attributes (i.e. update of RD_PATH) will be performed as part of ″operational processing″. h4 id=defxmp.Simple Default Policy

Among the concerns about configuring administrative policy is ease of configuration for the majority of domains which may have very simple policy. A simple policy configuration must include a preference statement:

(Assign a preference to all routes based on the number of hops.)

```
PREF {nlri_any} / .*/ = 255 - hopcount();
```

If the routing domain will carry transit traffic, then the following minimal aggregation and distribution statements are also needed:

(Automatic aggregation to reduce the amount of information.)

```
AGGR {nlri_any} / .*/ = auto;
```

(Select all routes for distribution; no modifications.)

```
DIST {nlri_any} / .*/ = select_on;
```

This illustrates that the configuration of the policy information base does not necessarily have to be extensive or complex. A complex configuration will be the case only to the extent that the domain's administrative policy has extensive requirements and specifications.