Title: An Example Policy Syntax for IDRP

Source: IBM

IBM has generated the attached example of a syntax that can specify the policies used within an implementation of IDRP (DIS 10747). We welcome discussion on what should be done with this material—for example, since it is informative in nature, it might be worthwhile to consider recommending that it be processed as a type 3 technical report.

IDRP Policy Configuration Specification
= =

This annex defines and describes one possible syntax, and the associated semantics, for a policy configuration language for the ISO IDRP protocol. Section 1 provides an overview of the policy configuration. Section 2 discusses the semantics of the each type of policy statement. Section 3 defines a BNF for a policy information base configuration language, and section 4 provides a simple example of the use of this policy configuration specification.

Alternative syntaxes with equivalent richness of functionality are not precluded. Other mechanisms may be needed to provide a fully functional configuration language.

[1.0] 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) The route preference section assigns a preference value to incoming routes; this is the "local selection policy" regarding routes. These policies determine which routes in the RIB-INs are selected for the LOC-RIB.

2) The route aggregation section 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) The route distribution section 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.

[2.0] Policy Statements

Each of the three policy subsections is comprised of a list of "policy statements" which serve to express the domain's policy. While the policy statements of each section are different, they all include a "route pattern" (which serves as a template for matching route attributes) and some "actions" which are to be associated with any matched route. Using these "match+action" pairs, a domain administrator can express the administrative policy for a routing domain.

[2.1] 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>]
```

<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 based on the presence/absence of the EXT_INFO attribute in a route. These tokens are optional; if not present, the default match is "idrp".

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

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

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

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

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

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

The <preference value expression> is an integer arithmetic expression with operators '+', '-', '*', '/', and (similar to the C language) a conditional operator '?'. The basic operands are constants, or pre-defined functions which return values based on the attributes of a route, e.g. hop_count(), 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.

[2.2] 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.]

[2.3] 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″.

[3.0] 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 section 2.0 above.

```
[3.1] 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>]

                              '=' <value>

## All of the symbols used by the <p_stmt> are also used in other
## places, and are defined in section [3.4] Common BNF Symbols.


[3.2] 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"


[3.3] 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.
## These 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> ′)′
```

[3.4] Common BNF Symbols

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

[3.4.1] BNF: Route Attribute Matching Templates

```
## 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;
## if priority attribute is present, or if it is not.
## If <empty>, default is ″priority_any″.


[3.4.2] BNF: Numerical Expressions
----------------------------------

<value> ::=
     <integer>     |
     <att_value>    |
     ′(′ <value> ′)′ |
     <value> <integer_op> <value> |
     <case_statement>

<case_statement> ::=  ′(′ <case_cond> ′?′ <value> ′:′ <value> ′)′

<att_value> ::=

     ″hopcount″   ″()″             |  ## rd_hop_count
     ″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.
```

```
[3.4.3] BNF: Conditional Specification
-------------------------------------

## CONDITIONS -- 3 (related) types:
##
## 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
 #
 ## 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>  |
               ″next_hop_any″ <local_snpa>
  #
  ## Can attempt to match NEXT_HOP attribute against ″any″,
  ## a set of BISs (NET prefix),
  ## a particular BIS and optionally specific interfaces.
  ## Also can match routes against local interface over
  ## which route was received.


[3.4.4] BNF: 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 rdi,
  ## 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>

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>

[4.0] 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.

[4.1] Transit Domain 3

Example policies of transit domain RD #3:

A) RD #3 only accepts IDRP originated routes. It supports two sets of distinguished RIB_ATTs: the default set (no distinguished attributes) and the set having only the CAPACITY QOS attribute.

B) Routes with CAPACITY QOS must travel via RD#6; CAPACITY must be greater than 15.

C) For routes with no distinguished attributes, prefer routes through transit domain 1, however preference should be given to routes with short paths; large hop counts on a route via RD#1 may cause a shift to another transit domain.

D) CAPACITY QOS routes are only offered to some domains (RDs #5, #8), and are restricted from being propagated further (i.e. via the DIST_LIST_INCL attribute).

E) All routes with no distinguished attributes are re-distributed to every neighbor RD; hierarchical recording is desired to limit distribution of all of these routes (the specific RDC membership information is irrelevant for this example).

F) Any route (default or CAPACITY) which pass through transit RD#9 (not pictured) can only be redistributed to some domains (RD#2, RD#5, RD#8).

G) All routes carrying NLRI of the address space controlled by domain #3 (XX:YY:3:*) will be aggregated (regardless 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).

[4.2] Policy Configuration Illustration

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();};
```

[4.3] 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.

[4.3.1] 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).

[4.3.2] 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).

[4.3.3] 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).

[4.4] 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) hop_count(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) hop_count(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) hop_count(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) hop_count(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) hop_count(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".

[4.5] 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 - hop_count();
```

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.