IDPR Working Group
Internet Draft

R.A. Woodburn, Editor
SAIC
July 1991

Definitions of Managed Objects for the Inter-Domain Policy Routing Protocol (Version 1)

1. Status of this Memo

This draft document will be submitted to the RFC editor as an experimental extension to the SNMP MIB. Distribution of this memo is unlimited. Please send comments to the IDPR Working Group (idpr-wg@bbn.com).

2. Abstract

This memo defines an experimental portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP-based internets. In particular, it defines objects for managing the Inter-Domain Policy Routing Protocol [10,11].

This memo does not specify a standard for the Internet community. However, after experimentation, if sufficient consensus is reached in the Internet community, then a subsequent revision of this document may be incorporated into the Internet-standard MIB.

3. Historical Perspective

As reported in RFC 1052, IAB Recommendations for the Development of Internet Network Management Standards [1], a two-prong strategy for network management of TCP/IP-based internets was undertaken. In the short-term, the Simple Network Management Protocol (SNMP), defined in RFC 1067, was to be used to manage nodes in the Internet community. In the long-term, the use of the OSI network management framework was to be examined. Two documents were produced to define the management information: RFC 1065, which defined the Structure of Management Information (SMI), and RFC 1066, which defined the Management Information Base (MIB). Both of these documents were designed so as to be compatible with both the SNMP and the OSI network management framework.

This strategy was quite successful in the short-term: Internet-based network management technology was fielded, by both the research and commercial communities, within a few months. As a result of this, portions of the Internet community became network manageable in a timely fashion.

As reported in RFC 1109, Report of the Second Ad Hoc Network Management Review Group [2], the requirements of the SNMP and the OSI network management frameworks were more different than anticipated. As such, the requirement for compatibility between the SMI/MIB and both frameworks was suspended. This action permitted the operational network management framework, based on the SNMP, to respond to new operational needs in the Internet community by producing MIB-II.

In May of 1990, the core documents were elevated to "Standard Protocols" with "Recommended" status. As such, the Internet-standard network management framework consists of: Structure and Identification of Management Information for TCP/IP-based internets, RFC 1155 [3], which describes how managed objects contained in the MIB are defined; Management Information Base for Network Management of TCP/IP-based internets, which describes the managed objects contained in the MIB, RFC 1156 [4]; and, the Simple Network Management Protocol, RFC 1157 [5], which defines the protocol used to manage these objects. Consistent with the IAB directive to produce simple, workable systems in the short-term, the list of managed objects defined in the Internet-standard MIB was derived by taking only those elements which are considered essential. However, the SMI defined three extensibility mechanisms: one, the addition of new standard objects through the definitions of new versions of the MIB; two, the addition of widely-available but non-standard objects through the experimental subtree; and three, the addition of private objects through the enterprises subtree. Such additional objects can not only be used for vendor-specific elements, but also for experimentation as required to further the knowledge of which other objects are essential.

This memo defines extensions to the MIB using the second method. It contains definitions of managed objects used for experimentation. After experimentation, if sufficient consensus is reached in the Internet community, then a subsequent revision of this memo may be placed in the Internet-standard MIB.

4. Objects

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) [7] defined in the SMI. In particular, each object has a name, a syntax, and an encoding. The name is an object identifier, an administratively assigned name, which specifies an object type. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, we often use a textual string, termed the OBJECT DESCRIPTOR, to also refer to the object type.

The syntax of an object type defines the abstract data structure corresponding to that object type. The ASN.1 language is used for this purpose. However, the SMI [3] purposely restricts the ASN.1 constructs which may be used. These restrictions are explicitly made for simplicity.

The encoding of an object type is simply how that object type is represented using the object type's syntax. Implicitly tied to the notion of an object type's syntax and encoding is how the object type is represented when being transmitted on the network.

The SMI specifies the use of the basic encoding rules of ASN.1 [8], subject to the additional requirements imposed by the SNMP.

4.1. Format of Definitions

Section 6 contains the specification of all object types contained in this MIB module. The object types are defined using the conventions defined in the SMI, as amended by the extensions specified in [9].

5. Overview

The Inter-Domain Policy Routing Protocol (IDPR) is a routing protocol for use between Administrative Domains. The objective of IDPR is to construct routes between source and destination administrative domains that provide user traffic with the service requested within the constraints stipulated by the domains transited.

With IDPR, we introduce a new network-layer Internet protocol based on source-specified routing between administrative domains and a new Internet addressing structure based on an administrative domain hierarchy. However, for IDPR version 1, we recommend implementing a proper subset of the complete architecture proposed in [11]. The recommended subset provides the majority of the features of IDPR and comprises the virtual gateway protocol, the domain status distribution protocol, the route synthesis procedure, the path setup protocol, and the message forwarding procedure.

5.1 Domain Structure

The IDPR architecture has been designed to accommodate an Internet with tens of thousands of administrative domains collectively containing hundreds of thousands of local networks. Inter-domain policy routes are constructed using information about the policy attributes of, and the connectivity between, administrative domains. The intra-domain details, gateways and networks traversed, of an inter-domain policy route are the responsibility of intra-domain routing and are thus outside the scope of inter-domain routing.

An Administrative Domain (AD) is a collection of contiguous networks managed by a single administrative authority that places policy restrictions on transit traffic, defines policy requirements for locally-generated traffic, and selects the intra-domain addressing schemes and routing procedures. Each administrative domain has a unique identity within the Internet.

Virtual Gateways (VGs) are the only IDPR-recognized connecting points between adjacent administrative domains. Each virtual gateway is a collection of directly-connected policy gateways in two adjacent domains, whose existence has been sanctioned by the authorities in both domains. The domain authorities may agree to establish more than one virtual gateway between the two domains. For each such virtual gateway, the two authorities together assign a virtual gateway identifier, unique within the set of virtual gateways connecting the two domains. To produce a virtual gateway identifier unique within its domain, the domain authority concatenates the mutually assigned identifier together with the adjacent domain's identifier.

Policy Gateways (PGs) are the physical gateways within a virtual gateway. Each policy gateway enforces policy restrictions on transit traffic as stipulated by the domain's administrative authority. A single policy gateway may belong to multiple virtual gateways. Within a domain, two policy gateways are neighbors if they are in different virtual gateways. Within a virtual gateway, two policy gateways are peers if they are in the same domain and are adjacent if they are in different domains. Adjacent policy gateways are directly connected if they are the only Internet-addressable entities attached to the connecting medium. Note that this definition implies that not only point-to-point links but also networks may serve as direct connections between adjacent policy gateways. The authority for a given domain assigns to each of its policy gateways an identifier, unique within that domain.

5.2 IDPR Functions

Inter-domain policy routing comprises the following functions:

1. Collecting and distributing connectivity and policy information pertaining to transit domains.
2. Synthesizing and selecting policy routes based on the connectivity and policy information associated with the transit domains and on the service requirements associated with the source.
3. Setting up paths across the Internet, using the policy routes synthesized.
4. Forwarding messages across and between administrative domains along the established paths.
5. Maintaining databases of transit domain connectivity and policy, inter-domain policy routes, configured global information such as network-address/domain-identifier mappings, and configured local information such as the policy gateways within an administrative domain.

As described in [11], a source domain controls synthesis and selection of policy routes to destination domains, while intermediate domains on a specific source-selected policy route determine whether or not the route is consistent with their transit policies. The route synthesis procedure uses domain status information in the form of advertised transit policies and adjacencies, to generate policy routes between source and destination domains. The source then selects policy routes from those provided by route synthesis, according to its own service requirements. Policy routes require a path setup procedure during which policy gateways in intermediate domains verify whether or not they will carry traffic travelling along the path, and contingent upon route acceptance, form an association between the path identifier and the previous and next policy gateways on the path. Following path setup, intermediate policy gateways use the path identifier carried in data messages to forward data traffic along the path.

Several different entities are responsible for performing the IDPR functions. Policy gateways collect and distribute status information about their administrative domains, participate in path setup, and forward data messages along established paths. Path agents act on behalf of hosts to select policy routes and to set up and manage paths. Special-purpose servers maintain the routing databases which are distributed with partial redundancy throughout the Internet. Each special-purpose server within an administrative domain has a unique identifier, assigned by the domain authority. Route servers are responsible for both the domain status (connectivity and policy) database and the route database. Also, route servers synthesize policy routes using domain status information and source traffic requirements. Name servers are responsible for the domain-name/network-address/domain-identifier database. Configuration servers are responsible for databases of configured information that apply to policy gateways, path agents, and route servers in the given administrative domain and inform these entities of configuration changes.

Both route servers and name servers are organized hierarchically, where a server's position in the hierarchy determines the extent of its database. At the top are global servers that maintain information about all Internet domains; at the bottom are local servers that maintain information about a particular domain, its neighbors, and other frequently visited domains, this last type of information usually obtained from higher-level servers. Hierarchical database organization releases hosts and gateways from the burden of maintaining information about large portions of the Internet, most of which they will never use.

In IDPR version 1, each policy gateway performs all IDPR functions, including those of the path agent and the special-purpose servers. Aggregating all routing functions into policy gateways simplifies implementation; one need only install IDPR protocols in policy gateways. Moreover, it simplifies communication between routing functions, as all functions reside within each policy gateway. We also note that IDPR version 1 supports only a flat route server hierarchy; each policy gateway contains a global route server. We are presently investigating approaches to making efficient use of hierarchically-organized routing information.

Given the size of the current Internet (on the order of 100 administrative domains) and the type of policies supported in IDPR version 1 (access restrictions only), we believe that policy gateways have adequate processing and memory resources to synthesize policy routes and set up paths as well as to forward messages and participate in intra-domain routing. Meanwhile, we are developing autonomous route servers and configuration servers, separate from policy gateways, so that IDPR can accommodate larger numbers of administrative domains and policies in the future.

5.3 The IDPR MIB

These objects are used to control and manage an IDPR [11] implementation.

This MIB is broken into 5 sections. The first section describes system variables. The remaining sections describe tables containing parts of the IDPR database. These tables include:

1. An Entity Table listing all other IDPR entities known by the IDPR instantiation.
2. A VG Table listing all IDPR Virtual Gateways known by the IDPR instantiation.
3. A Path Table listing all IDPR paths which originate at, traverse, or terminate at this IDPR instantiation.
4. An Address Map Table which maps addresses at the network layer to IDPR paths.

[Editor's Note: It is also desirable to have a table of policies. However, representing policies is extremely complex and requires many parallel tables to detail all the information thay can be contained when MIB objects of simple types are used. One proposal is to define a table of policies that would contain the source of policy and then an Octet String that would consist of the policy in the IDPR protocol format. This would make things simple for SNMP agents, but would require the managers to be able to parse policies in this form in order for them to display policy information. The alternative is to have very complex tables with a great deal of duplicate information. Does anyone have problems with allowing a complex Octet String such as is suggested?]

6. Definitions

```
RFCxxxx-MIB DEFINITIONS ::= BEGIN

IMPORTS
        experimental, IpAddress, Counter,  TimeTicks
                FROM RFC1155-SMI
        OBJECT-TYPE
                FROM RFC-1212
```

```
-- This MIB module uses the extended OBJECT-TYPE macro as
-- defined in [9]
-- The following types  are included for enhancing  readability

-- The EntityId type is an octet string where the first two octets combine to form the AD
-- identifier (with a  range of 0..65535).  The  last two octets combine to  form the Entity
-- number of the Entity in the AD.

EntityId ::= Octet String  (Size (4))

-- The VGId type is an octet string where the first two octets combine to form the
-- identifier of the  adjacent AD to which this  VG connects (with a range  of 0..65535).
-- The last octet forms the VG number of the Virtual Gateway for that adjacent AD.

VGId ::= Octet String  (Size (3))

-- The PathId type is an octet string where the first two octets combine to form the
-- identifier of the AD of origin for the path.  The next two octets  combine to form the PG
-- number of the  originating PG.  The last  four octects represent the path number for
-- that PG.

PathId ::= Octet String  (Size (8))

-- The BitField type is an octet string of four octets.  The first octet contains bit positions
-- 31-24.  The second octect contains bit positions  23-16.  The third octect  contains bit
-- positions 15-8.  The fourth octet contains bit  positions 7-0.  The values  of the octets
-- depend upon the bits that are set that are being represented.

BitField ::= Octet String  (Size (4))
```

6.1 IDPR System Object Definitions

```
idpr            OBJECT IDENTIFIER ::= { experimental 28 }
```

```
idprId          OBJECT-TYPE
        SYNTAX       EntityId
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The identifier of the  entity"
        ::= { idpr 1 }

idprType        OBJECT-TYPE
        SYNTAX       BitField
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The entity type, RS, or PG, or both.
                        Bit position 0  PG flag
                        Bit position 1  RS flag"
        ::= { idpr 2 }


idprADRep       OBJECT-TYPE
        SYNTAX       EntityId
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The identifier of the  AD Representive as viewed from this entity."
        ::= { idpr 3 }


idprUpTime      OBJECT-TYPE
        SYNTAX       TimeTicks
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The amount of time  since IDPR was last initialized in TimeTicks."
        ::= { idpr 4 }
```

```
--  [Editor's Note:  There are no sub-protocol  specific version numbers.  It is not clear just
--  what should be done.  Versions numbers are message type specific.  Each possible
--  message type in every sub-protocol has its  version number.  The Control protocol has
--  its own version number for its header, however, and it can have its own version related
--  objects.  If version numbers are given per message type, then the number of objects to
--  represent supported  versions will explode.  If  version numbers remain at the per
--  message level, then version objects may disappear from the mib.  If they are changed
--  to refer to  sub-protocols only, then these  objects make sense.  Does  anyone have
--  suggestions?  Is this use of version numbers a bad idea for the  IDPR protocol?  Note
--  that IDPR  encapsulation only has one message, so it is not affected by this problem.]

idprVGPVersions      OBJECT-TYPE
        SYNTAX       BitField
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The VGP versions supported by this PG.
                        Bit Position 0 ==       Version 1
                        Bit Position 1 ==       Version 2
                        ...and so on...
                        Bit Position n ==       Version n+1
                where n <= 31."
        ::= { idpr 5 }


idprVGPPer    OBJECT-TYPE
        SYNTAX       Integer
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The default value for  the UpDown protocol interval  (sec)."
        ::= { idpr 6 }


idprVGPPGIntOBJECT-TYPE
        SYNTAX       Integer
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The retransmission interval in seconds for acknowledged VGP  message
                types."
        ::= { idpr 7 }
```

```
idprVGPPGRet        OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of retransmissions to be sent for acknowledged VGP
                message types before timing them out"
        ::= { idpr 8 }


idprVGPSTSInt       OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The delay in seconds from the time a VG representative receives a PG
                connect message until it issues a PG status message for that VG"
        ::= { idpr 9 }


-- [Editor's Note: See the comment by "idprVGPVersions"]

idprUPDVersions     OBJECT-TYPE
        SYNTAX      BitField
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The versions of the Update protocol supported by this PG.
                        Bit Position 0 ==       Version 1
                        Bit Position 1 ==       Version 2
                        ...and so on...
                        Bit Position n ==       Version n+1
                where n <= 31."
        ::= { idpr 10 }


idprUPDReqInt       OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The retransmission interval in seconds for update requests"
        ::= { idpr 11 }
```

```
idprUPDReqRet       OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of retransmissions for update requests before  timing out the
                request"
        ::= { idpr 12 }


idprUPDPer    OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The interval in seconds  between periodic configuration   updates."
        ::= { idpr 13 }


idprUPDInt    OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The retransmission interval in seconds for configuration and dynamic
                updates at the time they are sent."
        ::= { idpr 14 }


idprUPDRet    OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of retransmissions for configuration and dynamic updates
                from the time of the first transmission until the  update is timed out."
        ::= { idpr 15 }


idprUPDTimNew       OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The maximum wait time in seconds before generating a 'correcting'
                update as a result of receiving an old update."
        := { idpr 16 }
```

```
idprUPDLif    OBJECT-TYPE
        SYNTAX       Integer
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The lifetime of received  updates in seconds."
        ::= { idpr 17 }


--  [Editor's Note:  See the comment by  "idprVGPVersions"]

idprSetupVersions      OBJECT-TYPE
        SYNTAX       BitField
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The versions of the  Setup protocol supported  by this PG.
                        Bit Position 0 ==        Version 1
                        Bit Position 1 ==        Version 2
                        ...and so on...
                        Bit Position n ==        Version n+1
                where n <= 31."
        ::= { idpr 18 }


idprSetupInt    OBJECT-TYPE
        SYNTAX       Integer
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The retransmission interval for setup messages in seconds"
        ::= { idpr 19 }


idprSetupRet   OBJECT-TYPE
        SYNTAX       Integer
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of retransmissions for setup messages before  they are timed
                out"
        ::= { idpr 20 }
```

```
-- [Editor's Note: See the comment by "idprVGPVersions"]

idprRSVersions       OBJECT-TYPE
        SYNTAX       BitField
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The versions of the Route Synthesis query protocol supported by this
                RS/PG.
                        Bit Position 0 ==       Version 1
                        Bit Position 1 ==       Version 2
                        ...and so on...
                        Bit Position n ==       Version n+1
                where n <= 31."
        ::= { idpr 21 }


idprControlVersions  OBJECT-TYPE
        SYNTAX       BitField
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The versions of the Control protocol supported by this RS/PG.
                        Bit Position 0 ==       Version 1
                        Bit Position 1 ==       Version 2
                        ...and so on...
                        Bit Position n ==       Version n+1
                where n <= 31."
        ::= { idpr 22 }


idprDataVersions     OBJECT-TYPE
        SYNTAX       BitField
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The versions of the encapsulation protocol supported by this PG.
                        Bit Position 0 ==       Version 1
                        Bit Position 1 ==       Version 2
                        ...and so on...
                        Bit Position n ==       Version n+1
                where n <= 31."
        ::= { idpr 23 }
```

```
idprDataErrs   OBJECT-TYPE
        SYNTAX       Counter
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  messages received which  could not be processed."
        ::= { idpr 24 }


idprDataUnkPaths      OBJECT-TYPE
        SYNTAX       Counter
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  Data messages received by this entity which
                referenced and unknown path."
        ::= { idpr 26 }
```

6.2 IDPR Entity Table Object Definitions

```
idprEntityTable        OBJECT-TYPE
        SYNTAX       SEQUENCE OF  IdprEntityEntry
        ACCESS       not-accessible
        STATUS       mandatory
        DESCRIPTION
                "The table of other  known Entities"
        ::= { idpr 27 }


idprEntityEntry        OBJECT-TYPE
        SYNTAX       IdprEntityEntry
        ACCESS       not-accessible
        STATUS       mandatory
        DESCRIPTION
                "Information for a single Entity"
        ::= { idprEntityTable 1  }
```

```
IdprEntityEntry ::= SEQUENCE {
        idprEntityTabId
                EntityId,
        idprEntityFlags
                BitField,
        idprEntityTabTrans
                Counter,
        idprEntityTabVGPVersion
                BitField,
        idprEntityTabVGPUpDownPer
                Integer,
        idprEntityTabVGPUpDownIns
                Counter,
        idprEntityTabVGPUpDownOuts
                Counter,
        idprEntityTabVGPPGStatIns
                Counter,
        idprEntityTabVGPPGStatOuts
                Counter,
        idprEntityTabVGPPGConnIns
                Counter,
        idprEntityTabVGPPGConnOuts
                Counter,
        idprEntityTabVGPVGStatIns
                Counter,
        idprEntityTabVGPVGStatOuts
                Counter,
        idprEntityTabVGPVGConnIns
                Counter,
        idprEntityTabVGPVGConnOuts
                Counter,
        idprEntityTabVGPUnkVersions
                Counter,
        idprEntityTabUPDVersion
                BitField,
        idprEntityTabUPDConfigIns
                Counter,
        idprEntityTabUPDConfigOuts
                Counter,
        idprEntityTabUPDDynamicIns
                Counter,
        idprEntityTabUPDDynamicOuts
                Counter,
        idprEntityTabUPDReqIns
                Counter,
        idprEntityTabUPDReqOuts
                Counter,
        idprEntityTabUPDUnkVersions
                Counter,
        idprEntityTabSetupVersion
                BitField,
        idprEntityTabSetupSetupIns
```

```
            Counter,
        idprEntityTabSetupSetupOuts
            Counter,
        idprEntityTabSetupTeardownIns
            Counter,
        idprEntityTabSetupTeardownOuts
            Counter,
        idprEntityTabSetupRefuseIns
            Counter,
        idprEntityTabSetupRefuseOuts
            Counter,
        idprEntityTabSetupAcceptIns
            Counter,
        idprEntityTabSetupAcceptOuts
            Counter,
        idprEntityTabSetupErrorIns
            Counter,
        idprEntityTabSetupErrorOuts
            Counter,
        idprEntityTabSetupUnkVersions
            Counter,
        idprEntityTabRSVersion
            BitField,
        idprEntityTabRSRtReqIns
            Counter,
        idprEntityTabRSRtRespOuts
            Counter,
        idprEntityTabRSUnreachs
            Counter,
        idprEntityTabRSUnkVersions
            Counter,
        idprEntityTabControlVersion
            Counter,
        idprEntityTabControlIns
            Counter,
        idprEntityTabControlOuts
            Counter,
        idprEntityTabControlRetries
            Counter,
        idprEntityTabControlAckIns
            Counter,
        idprEntityTabControlAckOuts
            Counter,
        idprEntityTabControlNakIns
            Counter,
        idprEntityTabControlNakOuts
            Counter,
        idprEntityTabControlTimeouts
            Counter
        idprEntityTabControlUnkVersions
            Counter
        idprEntityTabControlUnkAuths
```

```
                Counter
        idprEntityTabControlUnkProtos
                Counter
        idprEntityTabControlBadAuths
                Counter
        idprEntityTabControlBadLengths
                Counter
        idprEntityTabControlBadTimes
                Counter
}


idprEntityTabId         OBJECT-TYPE
        SYNTAX          EntityId,
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
                "The AD,id pair of this entity."
        ::= { idprEntityEntry 1  }


idprEntityFlags         OBJECT-TYPE
        SYNTAX          BitField
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
                "The type of entity and its status.
                        Bit Position 0 == Status - its presence indicates  the entity is UP, its
                                absence indicates that it  is DOWN.
                        Bit Position 1 == PG - the entity is acting as a PG
                        Bit Position 2 == RS - the entity is acting as a RS
                        Bit Position 3 == Peer - the entity is a  peer PG
                        Bit Position 4 ==  Neighbor - the entity is a neighbor PG
                        Bit Position 5 ==  Adjacent - the entity is  an adjacent PG"
        ::= { idprEntityEntry 2  }


idprEntityTabStateTrans         OBJECT-TYPE
        SYNTAX          Counter,
        ACCESS          read-only
        STATUS          mandatory
        DESCRIPTION
                "The number of times  the PG has transitioned state."
        ::= { idprEntityEntry 3  }
```

```
-- [Editor's Note:  See the comment by  "idprVGPVersions"]

idprEntityTabVGPVersion     OBJECT-TYPE
        SYNTAX      BitField,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The version of the VGP protocol being spoken with  this PG, if any.
                        Bit Position 0 ==       Version 1
                        Bit Position 1 ==       Version 2
                        ...and so on...
                        Bit Position n ==       Version n+1
                where n <= 31."
        ::= { idprEntityEntry 4  }


idprEntityTabVGPUpDownPer       OBJECT-TYPE
        SYNTAX      Integer,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The negotiated interval for Up/Down messages if any."
        ::= { idprEntityEntry 5  }


idprEntityTabVGPUpDownIns       OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Up/Down  messages received from the PG."
        ::= { idprEntityEntry 6  }


idprEntityTabVGPUpDownOuts      OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Up/Down  messages  sent to the PG."
        ::= { idprEntityEntry 7  }


idprEntityTabVGPPGStatIns OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of PG  STATUS messages received from the PG"
        ::= { idprEntityEntry 8  }
```

```
idprEntityTabVGPPGStatOuts        OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of PG STATUS messages sent to the PG"
        ::= { idprEntityEntry 9  }


idprEntityTabVGPPGConnIns         OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of PG  Connect messages received from the PG"
        ::= { idprEntityEntry 10  }


idprEntityTabVGPPGConnOuts        OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of PG  Connect messages sent to the PG"
        ::= { idprEntityEntry 11  }


idprEntityTabVGPVGStatIns OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of VG STATUS messages received from the  PG"
        ::= { idprEntityEntry 12  }


idprEntityTabVGPVGStatOuts        OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of VG  STATUS messages sent to the  PG"
        ::= { idprEntityEntry 13  }
```

```
idprEntityTabVGPVGConnIns         OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of VG  Connect messages received from the PG"
        ::= { idprEntityEntry 14  }


idprEntityTabVGPVGConnOuts        OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of VG  Connect messages sent to the PG."
        ::= { idprEntityEntry 15  }


idprEntityTabVGPUnkVersions       OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of VGP  messages received from the PG which contained an
                invalid version number."
        ::= { idprEntityEntry 16  }


--  [Editor's Note:  See the comment by  "idprVGPVersions"]

idprEntityTabUPDVersion     OBJECT-TYPE
        SYNTAX       BitField,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The version of the  Update protocol being spoken with this Entity, if any."
                        Bit Position 0 ==        Version 1
                        Bit Position 1 ==        Version 2
                        ...and so on...
                        Bit Position n ==        Version n+1
                where n <= 31."
        ::= { idprEntityEntry 17  }


idprEntityTabUPDConfigIns  OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of Config  Updates received from the Entity."
        ::= { idprEntityEntry 18  }
```

```
idprEntityTabUPDConfigOuts         OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Config  Updates sent to the Entity."
        ::= { idprEntityEntry 19  }


idprEntityTabUPDDynamicIns         OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Dynamic  Updates received from the Entity."
        ::= { idprEntityEntry 20  }


idprEntityTabUPDDynamicOuts        OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Dynamic  Updates sent to the Entity."
        ::= { idprEntityEntry 21  }


idprEntityTabUPDReqIns      OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Requests  received from the Entity."
        ::= { idprEntityEntry 22  }


idprEntityTabUPDReqOuts    OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Requests  sent to the Entity."
        ::= { idprEntityEntry 23  }
```

```
idprEntityTabUPDUnkVersions       OBJECT-TYPE
        SYNTAX       Counter
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of Update  messages received from this  Entity which
                contained an invalid version number."
        ::= { idprEntityEntry 24  }


-- [Editor's Note:  See the comment by  "idprVGPVersions"]

idprEntityTabSetupVersion   OBJECT-TYPE
        SYNTAX       BitField,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The version of the  Setup protocol being spoken with this PG, if any.
                        Bit Position 0 ==       Version 1
                        Bit Position 1 ==       Version 2
                        ...and so on...
                        Bit Position n ==       Version n+1
                where n <= 31."
        ::= { idprEntityEntry 25  }


idprEntityTabSetupSetupIns  OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of Setup  messages received from the PG."
        ::= { idprEntityEntry 26  }


idprEntityTabSetupSetupOuts          OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of Setup  messages sent to the PG."
        ::= { idprEntityEntry 27  }


idprEntityTabSetupTeardownIns       OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of Teardown  messages received from the PG."
        ::= { idprEntityEntry 28  }
```

```
idprEntityTabSetupTeardownOuts    OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Teardown  messages sent to the PG."
        ::= { idprEntityEntry 29  }


idprEntityTabSetupRefuseIns OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Refuse  messages received from the PG."
        ::= { idprEntityEntry 30  }


idprEntityTabSetupRefuseOuts      OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Refuse  messages sent to the PG."
        ::= { idprEntityEntry 31  }


idprEntityTabSetupAcceptInsOBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Accept  messages received from the PG."
        ::= { idprEntityEntry 32  }


idprEntityTabSetupAcceptOuts      OBJECT-TYPE
        SYNTAX      Counter,
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of Accept  messages sent to the PG."
        ::= { idprEntityEntry 33  }
```

```
idprEntityTabSetupErrorIns   OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of Error  messages received from the PG."
        ::= { idprEntityEntry 34  }


idprEntityTabSetupErrorOuts OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of Setup  Error messages sent to the  PG."
        ::= { idprEntityEntry 35  }


idprEntityTabSetupUnkVersions       OBJECT-TYPE
        SYNTAX       Counter
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of Setup  protocol messages received from  this PG which
                contained an invalid version number."
        ::= { idprEntityEntry 36  }


--  [Editor's Note:  See the comment by  "idprVGPVersions"]

idprEntityTabRSVersion       OBJECT-TYPE
        SYNTAX       BitField,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The version of the  Route Server protocol being spoken with this  Entity,
                if any.
                        Bit Position 0 ==       Version 1
                        Bit Position 1 ==       Version 2
                        ...and so on...
                        Bit Position n ==       Version n+1
                where n <= 31."
        ::= { idprEntityEntry 37  }
```

```
idprEntityTabRSRtReqIns     OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of Route  Requests received from the Entity."
        ::= { idprEntityEntry 38  }


idprEntityTabRSRtRespOuts OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of positive  Route Responses sent to the  Entity."
        ::= { idprEntityEntry 39  }


idprEntityTabRSUnreaches    OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of negative  Route Responses sent to the  Entity."
        ::= { idprEntityEntry 40  }


idprEntityTabRSUnkVersions          OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of RS  protocol messages received from  this entity which
                contained an unknown version number."
        ::= { idprEntityEntry 41  }


idprEntityTabControlVersion OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The version of the  Control protocol being spoken with this Entity.
                        Bit Position 0 ==        Version 1
                        Bit Position 1 ==        Version 2
                        ...and so on...
                        Bit Position n ==        Version n+1
                where n <= 31."
        ::= { idprEntityEntry 42  }
```

```
idprEntityTabControlIns        OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  Control received from the Entity."
        ::= { idprEntityEntry 43  }


idprEntityTabControlOuts      OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  Control sent to the Entity."
        ::= { idprEntityEntry 44  }


idprEntityTabControlRetries  OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  Control retransmissions sent to the Entity."
        ::= { idprEntityEntry 45  }


idprEntityTabControlAckIns  OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  Control ACKs received from the Entity."
        ::= { idprEntityEntry 46  }


idprEntityTabControlAckOuts           OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  Control ACKs sent to the  Entity."
        ::= { idprEntityEntry 47  }
```

```
idprEntityTabControlNakIns  OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  Control NAKs received from the Entity."
        ::= { idprEntityEntry 48  }


idprEntityTabControlNakOuts          OBJECT-TYPE
        SYNTAX       Counter,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  Control NAKs sent to the  Entity."
        ::= { idprEntityEntry 49  }


idprEntityTabControlTimeouts         OBJECT-TYPE
        SYNTAX       Counter
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  Control Timeouts for messages sent to the Entity."
        ::= { idprEntityEntry 50  }


idprEntityTabControlUnkVersions    OBJECT-TYPE
        SYNTAX       Counter
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  Control messages received from this entity which
                contained an unknown control protocl version number."
        ::= { idprEntityEntry 51  }


idprEntityTabControlUnkAuths        OBJECT-TYPE
        SYNTAX       Counter
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The total number of  control protocol messages received  from this entity
                which contained an unrecognized authentication type."
        ::= { idprEntityEntry 52  }
```

```
idprEntityTabControlUnkProtos       OBJECT-TYPE
        SYNTAX      Counter
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The total number of  control protocol messages received  from this entity
                which contained an invalid IDPR sub-protocol number."
        ::= { idprEntityEntry 53  }


idprEntityTabControlBadAuths        OBJECT-TYPE
        SYNTAX      Counter
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The total number of  control protocol messages received  from this entity
                for which authentication failed."
        ::= { idprEntityEntry 54  }


idprEntityTabControlBadLength       OBJECT-TYPE
        SYNTAX      Counter
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The total number of  control protocol messages received  from this entity
                which contained invalid length fields."
        ::= { idprEntityEntry 55  }


idprEntityTabControlBadTime         OBJECT-TYPE
        SYNTAX      Counter
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The total number of  control protocol messages received  from this entity
                which contained bad timestamps."
        ::= { idprEntityEntry 56  }
```

6.3 IDPR VG Table Object Definitions

```
idprVGTable   OBJECT-TYPE
        SYNTAX      SEQUENCE OF IdprVGEntry
        ACCESS      not-accessible
        STATUS      mandatory
        DESCRIPTION
                "The table of known  VGs"
        ::= { idpr 28 }
```

```
idprVGEntry   OBJECT-TYPE
        SYNTAX       IdprVGEntry
        ACCESS       not-accessible
        STATUS       mandatory
        DESCRIPTION
                "Information for a single  VG."
        ::= { idprVGTable 1 }


IdprVGEntry ::= SEQUENCE {
        idprVGTabId
                VGId,
        idprVGTabFlags
                Counter,
        idprVGTabVGRep
                EntityId
}


idprVGTabId  OBJECT-TYPE
        SYNTAX       VGId,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The Adjacent AD,id pair  of this VG."
        ::= { idprVGEntry 1 }


idprVGTabFlags       OBJECT-TYPE
        SYNTAX       BitField,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "This is a string of bits whose value should be interpreted as the sum of
                powers of 2.  The values of the different fields are:
                        2^0     == 1    Status - its presence indicates the VG is UP,
                                        its absence indicates that it is DOWN.
                        2^1     == 2    MemberPG - this entity is a  member of this VG
                The values of the flags are summed to obtain the value of the integer.  For
                example, if the VG is UP (1) and is this  PG is a member of the VG (2) the
                value would be the 1 + 2, which is 3"
        ::= { idprVGEntry 2 }


idprVGTabVGRep    OBJECT-TYPE
        SYNTAX       EntityId
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The PG acting as the VG  representative for the VG as seen  by this PG."
        ::= { idprVGEntry 3 }
```

6.4 IDPR Path Table Object Definitions

```
idprPathTable OBJECT-TYPE
        SYNTAX       SEQUENCE OF IdprPathEntry
        ACCESS       not-accessible
        STATUS       mandatory
        DESCRIPTION
                "The table IDPR paths."
        ::= { idpr 29 }


idprPathEntry OBJECT-TYPE
        SYNTAX       IdprPathEntry
        ACCESS       not-accessible
        STATUS       mandatory
        DESCRIPTION
                "Information for a single  IDPR path."
        ::= { idprPathTable 1 }
```

```
IdprPathEntry ::= SEQUENCE {
        idprPathTabId
                PathId,
        idprPathTabADDst
                Integer
        idprPathTabPGNext
                EntityId,
        idprPathTabPGPrev
                EntityId,
        idprPathTabState
                Integer
        idprPathTabMsgs
                Counter,
        idprPathTabMsgsMax
                Integer,
        idprPathTabBytes
                Counter,
        idprPathTabBytesMax
                Integer,
        idprPathTabExpiration
                Integer,
        idprPathTabAuth
                Integer,
        idprPathTabVersion
                Integer,
        idprPathTabTos
                Octet String (Size (8)),
        idprPathTabUCI
                Octet String (Size (8)),
        idprPathTabErrs
                Counter,
        idprPathTabUnkProtos
                Counter,
        idprPathTabBadLengths
                Counter,
        idprPathTabBadAuths
                Counter,
}


idprPathTabId OBJECT-TYPE
        SYNTAX       PathId
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The unique identifier for this path."
        ::= { idprPathEntry 1 }
```

```
idprPathTabADDst    OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The final destination AD  for this path."
        ::= { idprPathEntry 2 }


idprPathTabPGNext   OBJECT-TYPE
        SYNTAX      EntityId
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The ID of the next hop PG."
        ::= { idprPathEntry 3 }


idprPathTabPGPrev   OBJECT-TYPE
        SYNTAX      EntityId
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The ID of the previous hop PG."
        ::= { idprPathEntry 4 }


idprPathTabState    OBJECT-TYPE
        SYNTAX      Integer {
                    idle (0),  -- the path is no longer  being used
                    setup (1),  -- a setup request has been sent
                    accept_wait (2),   -- an ACK has been  received
                    active (3)  -- the end-to-end accept has been received
        }
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The current state of  the path.."
        ::= { idprPathEntry 5 }


idprPathTabMsgs     OBJECT-TYPE
        SYNTAX      Counter
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of data  messages forwarded on this path."
        ::= { idprPathEntry 6 }
```

```
idprPathTabMsgsMax OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The maximum allowable number of data messages forwarded on this
                path."
        ::= { idprPathEntry 7 }


idprPathTabBytes    OBJECT-TYPE
        SYNTAX      Counter
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of bytes of data forwarded on this  path."
        ::= { idprPathEntry 8 }


idprPathTabBytesMax         OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The maximum allowable number of bytes of data  forwarded on this
                path."
        ::= { idprPathEntry 9 }


idprPathTabExpiration       OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The time at which this path will expire."
        ::= { idprPathEntry 10  }


idprPathTabAuth     OBJECT-TYPE
        SYNTAX      Integer
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The authentication used for this path."
        ::= { idprPathEntry 11  }
```

```
idprPathTabVersion   OBJECT-TYPE
        SYNTAX       Integer
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The version of the  encapsulation protocol expected for  this path.
                        Bit Position 0 ==       Version 1
                        Bit Position 1 ==       Version 2
                        ...and so on...
                        Bit Position n ==       Version n+1
                where n <= 31."
        ::= { idprPathEntry 12 }


idprPathTabTos       OBJECT-TYPE
        SYNTAX       Octet String (Size (8))
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The type of services  associated with this path.   Each octet in the string
                represents a different type of service supported."
        ::= { idprPathEntry 13 }


idprPathTabUCI       OBJECT-TYPE
        SYNTAX       Octet String (Size (8))
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The User Class Identifiers associated with this path.  Each octet of the
                string represents a different supported UCI."
        ::= { idprPathEntry 14 }


idprPathTabDataErrs  OBJECT-TYPE
        SYNTAX       Counter
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of packets  received which contained an error."
        ::= { idprPathEntry 15  }


idprPathTabDataUnkProtos   OBJECT-TYPE
        SYNTAX       Counter
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The number of packets received on this path which had an invalid
                protocol field."
        ::= { idprPathEntry 16  }
```

```
idprPathTabDataBadLengths OBJECT-TYPE
        SYNTAX      Counter
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of packets  received on this path which  had an invalid length
                field."
        ::= { idprPathEntry 17 }


idprPathTabDataBadAuths    OBJECT-TYPE
        SYNTAX      Counter
        ACCESS      read-only
        STATUS      mandatory
        DESCRIPTION
                "The number of data  packets received on this path which failed
                authentication."
        ::= { idprPathEntry 18 }
```

6.5 IDPR Address Mapping Table Object Definitions

```
idprAddrMapTable    OBJECT-TYPE
        SYNTAX      SEQUENCE OF  IdprAddrMapEntry
        ACCESS      not-accessible
        STATUS      mandatory
        DESCRIPTION
                "The address mapping table from source and destination  addresses to a
                particular IDPR path."
        ::= { idpr 30 }


idprAddrMapEntry    OBJECT-TYPE
        SYNTAX      IdprAddrMapEntry
        ACCESS      not-accessible
        STATUS      mandatory
        DESCRIPTION
                "Information for a single IDPR Address Map."
        ::= { idprPathTable 1 }
```

```
IdprAddrMapEntry ::= SEQUENCE {
        idprAMTSrc
                IpAddress,
        idprAMTSrcMask
                IpAddress,
        idprAMTDst
                IpAddress,
        idprAMTDstMask
                IpAddress,
        idprAMTPathId
                PathId,
        idprAMTState
                Integer
}


idprAMTSrc   OBJECT-TYPE
        SYNTAX       IpAddress,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The source address for  the map"
        ::= { idprAddrMapEntry 1  }


idprAMTSrcMask      OBJECT-TYPE
        SYNTAX       IpAddress,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The source address mask  for the map"
        ::= { idprAddrMapEntry 2  }


idprAMTDst   OBJECT-TYPE
        SYNTAX       IpAddress,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The destination address for the map"
        ::= { idprAddrMapEntry 3  }


idprAMTDstMask      OBJECT-TYPE
        SYNTAX       IpAddress,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The destination address mask for the  map"
        ::= { idprAddrMapEntry 4  }
```

```
idprAMTPathId        OBJECT-TYPE
        SYNTAX       PathId,
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The path this mapping  uses."
        ::= { idprAddrMapEntry 5  }


idprAMTState OBJECT-TYPE
        SYNTAX       Integer{
                     idle (0),  -- no request pending, no path assigned
                     waiting (1),  -- request pending, no path  assigned
                     active (2)  -- path assigned
        }
        ACCESS       read-only
        STATUS       mandatory
        DESCRIPTION
                "The state of this  mapping."
::= { idprAddrMapEntry 6  }

--  [Editor's Note:  It is also desirable to have a reference to the  policy term that was used
--  from the source in the generation of this path, perhaps even the complete route along
--  with the associated policy information.  However, this has the same problem  that
--  policies themselves  do of being a very complex bit of complex information.   It would
--  be possible to do this through a bunch  of tables, but it would be useful if the policy
--  identifiers could be listed in some binary  format in an Octet String.   Is this a "bad"
--  thing to do?  Would it be possible to have the same form as is used in the Octet  Strings
--  for TOS and  UCI?]
```

7. Acknowledgements

We would like to acknowledge the assistance of all the members of the IDPR Working Group and particularly the following individuals :

Ken Carlberg, SAIC
Mark Sleeper, Sparta
Martha Steenstrup, BBN

8. References

[1] V. Cerf, *IAB Recommendations for the Development of Internet Network Management Standards*, Internet Working Group Request for Comments 1052. Network Information Center, SRI International, Menlo Park, California, April 1988.

[2] V. Cerf, *Report of the Second Ad Hoc Network Management Review Group*, Internet Working Group Request for Comments 1109. Network Information Center, SRI International, Menlo Park, California, August 1989.

[3] M.T. Rose and K. McCloghrie, *Structure and Identification of Management Information for TCP/IP-based internets*, Internet Working Group Request for Comments 1155. Network Information Center, SRI International, Menlo Park, California, May 1990.

[4] K. McCloghrie and M.T. Rose, *Management Information Base for Network Management of TCP/IP-based internets*, Internet Working Group Request for Comments 1156. Network Information Center, SRI International, Menlo Park, California, May 1990.

[5] J.D. Case, M.S. Fedor, M.L. Schoffstall, and J.R. Davin, *Simple Network Management Protocol*, Internet Working Group Request for Comments 1157. Network Information Center, SRI International, Menlo Park, California, May 1990.

[6] M.T. Rose (editor), *Management Information Base for Network Management of TCP/IP-based internets*, Internet Working Group Request for Comments 1158, Network Information Center, SRI International, Menlo Park, California, May 1990.

[7] *Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1)*, International Organization for Standardization. International Standard 8824, December 1987.

[8] *Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Notation One (ASN.1)*, International Organization for Standardization. International Standard 8825, December 1987.

[9] M.T. Rose, K. McCloghrie (editors), *Towards Concise MIB Definitions*, Internet Working Group Request for Comments 1212, Network Information Center, SRI International, Menlo Park, California, September 1990.

[10] M. Lepp and M. Steenstrup, *An Architecture for Inter-Domain Policy Routing*, Internet Draft, July 1990.

[11] ORWG, *Inter-Domain Policy Routing Protocol Specification and Usage*, Internet Draft, July 1991.

9. Editor's Adress

IDPR Working Group
c/o Robert A. Woodburn
SAIC CSEIC
8619 Westwood Center Drive
Vienna, VA 22182
(703) 734-9000 or (703) 448-0210

EMAIL: idpr-wg@bbn.com or woody@cseic.saic.com