                   A Simple Gateway Monitoring Protocol

1. Status of this Memo

   This document is being distributed to members of the Internet
   community in order to solicit their reactions to the proposals
   contained in it. While the issues discussed may not be directly
   relevant to the research problems of the Internet, they may be
   interesting to a number of researchers and implementors.

   This memo defines a simple application-layer protocol by which
   management information for a gateway may be inspected or altered by
   logically remote users.

   This proposal is intended only as an interim response to immediate
   gateway monitoring needs while work on more elaborate and robust
   designs proceeds with the care and deliberation appropriate to that
   task. Accordingly, long term use of the mechanisms described here
   should be seriously questioned as more comprehensive proposals emerge
   in the future. Distribution of this memo is unlimited.

2. Protocol Design Strategy

   The proposed protocol is shaped in large part by the desire to
   minimize the number and complexity of management functions realized
   by the gateway itself. This goal is attractive in at least four
   respects:

   (1) The development cost for gateway software necessary to
        support the protocol is accordingly reduced.

   (2) The degree of management function that is remotely
        supported is accordingly increased, thereby admitting
        fullest use of internet resources in the management task.

(3) The degree of management function that is remotely
     supported is accordingly increased, thereby imposing the
     fewest possible restrictions on the form and sophistication
     of management tools.

(4) A simplified set of management functions is easily
     understood and used by developers of gateway management
     tools.

A second design goal is that the functional paradigm for monitoring
and control be sufficiently extensible to accommodate additional,
possibly unanticipated aspects of gateway operation.

A third goal is that the design be, as much as possible, independent
of the architecture and mechanisms of particular hosts or particular
gateways.

Consistent with the foregoing design goals are a number of decisions
regarding the overall form of the protocol design.

One such decision is to model all gateway management functions as
alterations or inspections of various parameter values. By this
model, a protocol entity on a logically remote host (possibly the
gateway itself) interacts with a protocol entity resident on the
gateway in order to alter or retrieve named portions (variables) of
the gateway state. This design decision has at least two positive
consequences:

(1) It has the effect of limiting the number of essential
     management functions realized by the gateway to two: one
     operation to assign a value to a specified configuration
     parameter and another to retrieve such a value.

(2) A second effect of this decision is to avoid introducing
     into the protocol definition support for imperative
     management commands: the number of such commands is in
     practice ever-increasing, and the semantics of such
     commands are in general arbitrarily complex.

The exclusion of imperative commands from the set of explicitly
supported management functions is unlikely to preclude any desirable
gateway management operation. Currently, most gateway commands are
requests either to set the value of some gateway parameter or to
retrieve such a value, and the function of the few imperative
commands currently supported is easily accommodated in an
asynchronous mode by this management model. In this scheme, an
imperative command might be realized as the setting of a parameter
value that subsequently triggers the desired action.

A second design decision is to realize any needed authentication
functionality in a distinct protocol layer that provides services to
the monitoring protocol itself. The most important benefit of this
decision is a reduction in the complexity of the individual protocol
layers - thereby easing the task of implementation.

Consistent with this layered design strategy is a third design
decision that the identity of an application protocol entity is known
to its peers only by the services of the underlying authentication
protocol. Implicit in this decision is a model of access control by
which access to variables of a gateway configuration is managed in
terms of the association between application entities and sessions of
the authentication protocol. Thus, multi-level access to gateway
variables is supported by multiple instances of the application
protocol entity, each of which is characterized by:

(1) the set of gateway variables known to said entity,

(2) the mode of access (READ-ONLY or READ-WRITE) afforded to
     said set of variables, and

(3) the authentication protocol session to which belong the
     messages sent and received by said entity.

A fourth design decision is to adopt the conventions of the CCITT
X.409 recommendation [1] for representing the information exchanged
between protocol entities. One cost of this decision is a modest
increase in the complexity of the protocol implementation. One
benefit of this decision is that protocol data are represented on the
network in a machine-independent, widely understood, and widely
accepted form. A second benefit of this decision is that the form of
the protocol messages may be concisely and understandably described
in the X.409 language defined for such purposes.

A fifth design decision, consistent with the goal of minimizing
gateway complexity, is that the variables manipulated by the protocol
assume only integer or octet string type values.

A sixth design decision, also consistent with the goal of minimizing
gateway complexity, is that the exchange of protocol messages
requires only an unreliable datagram transport, and, furthermore,
that every protocol message is entirely and independently
representable by a single transport datagram. While this document
specifies the exchange of protocol messages via the UDP protocol [2],
the design proposed here is in general suitable for use with a wide
variety of transport mechanisms.

A seventh design decision, consistent with the goals of simplicity and extensibility, is that the variables manipulated by the protocol are named by octet string values. While this decision departs from the architectural traditions of the Internet whereby objects are identified by assigned integer values, the naming of variables by octet strings affords at least two valuable benefits. Because the set of octet string values constitutes a variable name space that, as convenient, manifests either flat or hierarchical structure,

(1)   a single, simple mechanism can provide both random access to individual variables and sequential access to semantically related groups of variables, and

(2)   the variable name space may be extended to accommodate unforeseen needs without compromising either the relationships among existing variables or the potential for further extensions to the space.

An eighth design decision is to minimize the number of unsolicited messages required by the protocol definition. This decision is consistent with the goal of simplicity and motivated by the desire to retain maximal control over the amount of traffic generated by the network management function - even at the expense of additional protocol overhead. The strategy implicit in this decision is that the monitoring of network state at any significant level of detail is accomplished primarily by polling for appropriate information on the part of the monitoring center. In this context, the definition of unsolicited messages in the protocol is confined to those strictly necessary to properly guide a monitoring center regarding the timing and focus of its polling.

3.   The Gateway Monitoring Protocol

The gateway monitoring protocol is an application protocol by which the variables of a gateway's configuration may be inspected or altered.

Communication among application protocol entities is by the exchange of protocol messages using the services of the authentication protocol described elsewhere in this document. Each such message is entirely and independently represented by a single message of the underlying authentication protocol. An implementation of this protocol need not accept protocol messages whose length exceeds 484 octets.

The form and function of the four message types recognized by a protocol entity is described below. The type of a given protocol message is indicated by the value of the implicit type tag for the

data structure that is represented by said message according to the
conventions of the CCITT X.409 recommendation.

3.1. The Get Request Message Type

The form of a message of Get Request type is described below in the
language defined in the CCITT X.409 recommendation:

```
var_value_type              ::=       CHOICE {

                                      INTEGER,
                                      OCTET STRING

                                        }

var_name_type               :=        OCTET STRING

var_op_type                 ::=       SEQUENCE {

                                var_name              var_name_type,
                                var_value             var_value_type

                                }

var_op_list_type            ::=       SEQUENCE OF var_op_type

error_status_type           ::=       INTEGER {

                                gmp_err_noerror       (0),
                                gmp_err_too_big       (1),
                                gmp_err_nix_name      (2),
                                gmp_err_bad_value     (3)

                                }

error_index_type            ::=       INTEGER

request_id_type             ::=       INTEGER

get_req_message_type        ::=       [ APPLICATION 1 ] IMPLICIT

                            SEQUENCE {

                                request_id            request_id_type,
                                error_status          error_status_type,
                                error_index           error_index_type,
                                var_op_list           var_op_list_type
```

```
            }
```

   Upon receipt of a message of this type, the receiving entity responds
   according to any applicable rule in the list below:

   (1)   If, for some var_op_type component of the received message, the
         value of the var_name field does not lexicographically precede
         the name of some variable known to the receiving entity, then
         the receiving entity sends to the originator of the received
         message a message of identical form except that the indicated
         message type is Get Response, the value of the error_status
         field is gmp_err_nix_name, and the value of the error_index
         field is the unit-based index of said var_op_type component in
         the received message.

   (2)   If the size of the Get Response type message generated as
         described below would exceed the size of the largest message
         for which the protocol definition requires acceptance, then the
         receiving entity sends to the originator of the received message
         a message of identical form except that the indicated message
         type is Get Response, the value of the error_status field is
         gmp_err_too_big, and the value of the error_index field is zero.

   If none of the foregoing rules apply, then the receiving entity sends
   to the originator of the received message a Get Response type message
   such that, for each var_op_type component of the received message, a
   corresponding component of the generated message represents the name
   and value of that variable whose name is, in the lexicographical
   ordering of the names of all variables known to the receiving entity
   together with the value of the var_name field of the given component,
   the immediate successor to that value. The value of the error_status
   field of the generated message is gmp_err_noerror and the value of
   the error_index field is zero. The value of the request_id field of
   the generated message is that for the received message.

   Messages of the Get Request type are generated by a protocol entity
   only at the request of the application user.

3.2. The Get Response Message Type

   The form of messages of this type is identical to that of Get Request
   type messages except for the indication of message type. In the CCITT
   X.409 language,

   get_rsp_message_type   ::=   [APPLICATION 2] IMPLICIT

                            SEQUENCE {

```
                                  request_id                request_id_type,
                                  error_status              error_status_type,
                                  error_index               error_index_type,
                                  var_op_list               var_op_list_type

                               }
```

   The response of a protocol entity to a message of this type is to
   present its contents to the application user.

   Messages of the Get Response type are generated by a protocol entity
   only upon receipt of Set Request or Get Request type messages as
   described elsewhere in this document.

3.3. The Trap Request Message Type

   The form of a message of this type is described below in the language
   defined in the CCITT X.409 recommendation:

```
   val_list_type              ::=      SEQUENCE OF var_value_type

   trap_type_type             ::=      INTEGER

   trap_req_message_type      ::=      [ APPLICATION 3 ] IMPLICIT

                               SEQUENCE {

                               trap_type                 trap_type_type,
                               val_list                  val_list_type

                               }
```

   The response of a protocol entity to a message of this type is to
   present its contents to the application user.

   Messages of the Trap Request type are generated by a protocol entity
   only at the request of the application user.

   The significance of the val_list component of a Trap Request type
   message is implementation-specific.

   Interpretations for negative values of the trap_type field are
   implementation-specific. Interpretations for non-negative values of
   the trap_type field are defined below.

3.3.1. The Cold Start Trap Type

   A Trap Request type message for which the value of the trap_type

field is 0, signifies that the sending protocol entity is
reinitializing itself such that the gateway configuration or the
protocol entity implementation may be altered.

3.3.2. The Warm Start Trap Type

A Trap Request type message for which the value of the trap_type
field is 1, signifies that the sending protocol entity is
reinitializing itself such that neither the gateway configuration nor
the protocol entity implementation is altered.

3.3.3. The Link Failure Trap Type

A Trap Request type message for which the value of the trap_type
field is 2, signifies that the sending protocol entity recognizes a
failure in one of the communication links represented in the gateway
configuration.

3.3.4. The Authentication Failure Trap Type

A Trap Request type message for which the value of the trap_type
field is 3, signifies that the sending protocol entity is the
addressee of a protocol message that is not properly authenticated.

3.3.5. The EGP Neighbor Loss Trap Type

A Trap Request type message for which the value of the trap_type
field is 4, signifies that an EGP neighbor for whom the sending
protocol entity was an EGP peer has been marked down and the peer
relationship no longer obtains.

3.4. The Set Request Message Type

The form of messages of this type is identical to that of Get Request
type messages except for the indication of message type. In the
CCITT X.409 language:

```
set_req_message_type      ::=      [ APPLICATION 4 ] IMPLICIT

                          SEQUENCE {

                          request_id              request_id_type,
                          error_status            error_status_type,
                          error_index             error_index_type,
                          var_op_list             var_op_list_type

                          }
```

Upon receipt of a message of this type, the receiving entity responds according to any applicable rule in the list below:

(1)   If, for some var_op_type component of the received message, the value of the var_name field names no variable known to the receiving entity, then the receiving entity sends to the originator of the received message a message of identical form except that the indicated message type is Get Response, the value of the error_status field is gmp_err_nix_name, and the value of the error_index field is the unit-based index of said var_op_type component in the received message.

(2)   If, for some var_op_type component of the received message, the contents of the var_value field does not, according to the CCITT X.409 recommendation, manifest a type, length, and value that is consistent with that required for the variable named by the value of the var_name field, then the receiving entity sends to the originator of the received message a message of identical form except that the indicated message type is Get Response, the value of the error_status field is gmp_err_bad_value, and the value of the error_index field is the unit-based index of said var_op_type component in the received message.

(3)   If the size of the Get Response type message generated as described below would exceed the size of the largest message for which the protocol definition requires acceptance, then the receiving entity sends to the originator of the received message a message of identical form except that the indicated message type is Get Response, the value of the error_status field is gmp_err_too_big, and the value of the error_index field is zero.

If none of the foregoing rules apply, then for each var_op_type component of the received message, according to the sequence of such components represented by said message, the value represented by the var_value field of the given component is assigned to the variable named by the value of the var_name field of that component. The receiving entity sends to the originator of the received message a message of identical form except that the indicated message type is Get Response, the value of the error_status field is gmp_err_noerror, and the value of the error_index field is zero.

Messages of the Set Request type are generated by a protocol entity only at the request of the application user.

Recognition and processing of Set Request type frames is not required by the protocol definition.

4. The Authentication Protocol

   The authentication protocol is a session-layer protocol by which
   messages specified by a protocol user are selectively delivered to
   other protocol users. The protocol definition precludes delivery to
   a protocol user of any user message for which the protocol
   representation lacks a specified "authentic" form.

   Communication among authentication protocol entities is accomplished
   by the exchange of protocol messages, each of which is entirely and
   independently represented by a single UDP datagram. An
   authentication protocol entity responds to protocol messages received
   at UDP port 153 on the host with which it is associated.

   A half-session of the authentication protocol is, for any ordered
   pair of protocol users, the set of messages sent from the first user
   of the pair to the second user of said pair. A session of the
   authentication protocol is defined to be union of two complementary
   half-sessions of the protocol - that is, the set of messages
   exchanged between a given pair of protocol users. Associated with
   each protocol half-session is a triplet of functions:

   (1)   The authentication function for a given half-session is a
         boolean-valued function that characterizes the set of
         authentication protocol messages that are of acceptable,
         authentic form with respect to the set of all possible
         authentication protocol messages.

   (2)   The message interpretation function for a given half-
         session is a mapping from the set of authentication
         protocol messages accepted by the authentication function
         for said half-session to the set of all possible user
         messages.

   (3)   The message representation function for a given half-
         session is a mapping that is the inverse of the message
         interpretation function for said half-session.

   The association between half-sessions of the authentication protocol
   and triplets of functions is not defined in this document.

   The form and function of the single message type recognized by a
   protocol entity is described below. The type of a given protocol
   message is indicated by the value of the implicit type tag for the
   data structure that is represented by said message according to the
   conventions of the CCITT X.409 recommendation.

4.1. The Data Request Message Type

   Messages of this type are represented by a sequence of fields whose
   form and interpretation are described below.

4.1.1. The Message Length Field

   The Message Length field of a given Data Request message represents
   the length of said message as an unsigned, 16-bit, binary integer.
   This value is encoded such that more significant bits precede less
   significant bits in the order of transmission and includes the length
   of the Message Length field itself.

4.1.2. The Session ID Length Field

   The Session ID Length field of a given Data Request message
   represents the length, in octets, of the Session ID field of said
   message. This value is encoded as an unsigned, 8-bit, binary
   integer.

4.1.3. The Session ID Field

   The Session ID field of a given Data Request message represents the
   name of the protocol session to which said message belongs. The
   value of this field is encoded as asequence of octets whose length is
   the value of the Session ID Length field for said message.

4.1.4. The User Data Field

   The User Data field of a given Data Request message represents a
   message being passed from one protocol user to another. The value of
   this field is encoded according to conventions implicit in the
   message representation function for the appropriate half of the
   protocol session named by the value of the Session ID field for said
   message.

   Upon receipt of a Data Request type message, the receiving
   authentication protocol entity verifies the form of said message by
   application of the authentication function associated with its half
   of the session named by the value of the Session ID field in the
   received message. If the form of the received message is accepted as
   "authentic" by said function, then the user message computed by the
   application of the message interpretation function for said half-
   session to the value of the User Data field of the received message
   is presented to the protocol user together with an indication of the
   protocol session to which the received message belongs.

Otherwise, the message is discarded and an indication of the receipt
of an unauthenticated message is presented to the protocol user.

A message of this type is generated only at the request of the
protocol user to communicate a message to another user of the
protocol. Such a request specifies the user message to be sent as
well as the session of the authentication protocol to which said user
message belongs. The value of the Session ID field of the generated
message is the name of the session specified in the user request.
The value of the User Data field of the generated message is computed
by applying the message representation function for the appropriate
half of the specified session to the specified user message.

5. Variable Names

The variables retrieved or manipulated by the application protocol
are named by octet string values. Such values are represented in
this document in two ways:

(1) A variable name octet string may be represented
     numerically by a sequence of hexadecimal numbers, each of
     which denotes the value of the corresponding octet in
     said string.

(2) A variable name octet string may be represented
     symbolically by a character string whose form reflects
     the sequence of octets in said name while at the same
     time suggesting to a human reader the semantics of the
     named variable.

Variable name octet strings are represented symbolically according to
the following two rules:

(1) The symbolic character string representation of the
     variable name of zero length is the character string of
     zero length.

(2) The symbolic character string representation of a
     variable name of non-zero length n is the concatenation
     of the symbolic character string representation of the
     variable name formed by the first (n - 1) octets of the
     given name together with the underscore character ("_")
     and a character string that does not include the
     underscore character, such that the resulting character
     string is unique among the symbolic character string
     representations for all variable names of length n.

Thus, for example, the variable names represented numerically as:

```
01 01 01,
01 01 02,
01 02 01,
01 03 01 03 01,
01 03 01 03 02,
01 03 01 04 01, and
01 03 01 04 02
```

might be represented symbolically by the character strings:

```
_GW_version_id,
_GW_version_rev,
_GW_cfg_nnets,
_GW_net_if_type_net1,
_GW_net_if_type_net2,
_GW_net_if_speed_net1, and
_GW_net_if_speed_net2.
```

All variable names are terminated by an implementation specific octet
string of non-zero length. Thus, a complete variable name is not
specified for any of the variables defined in this document. Rather,
for each defined variable, some prefix portion of its name is
specified, with the understanding that the rightmost portion of its
name is specific to the protocol implementation.

Fullest exploitation of the semantics of the Get Request type message
requires that names for related variables be chosen so as to be
contiguous in the lexicographic ordering of all variable names
recognized by an application protocol entity. This principle is
observed in the naming of variables currently defined by this
document, and it should be observed as well for variables defined by
subsequent revisions of this document and for variables introduced by
particular implementations of the protocol.

A particular implementation of a protocol entity may present
variables in addition to those defined by this document, provided
that in no case will an implementation-specific variable be presented
as having a name identical to that for one of the variables defined
here. By convention, the names of variables specific to a particular
implementation share a common prefix that distinguishes said
variables from those defined in this document and from those that may
be presented by other implementations of an application protocol
entity. For example, variables specific to an implementation of this
protocol in version 1.3 of the Squeaky gateway product of the
Swinging Gateway company might have the names represented by:

```
                01 FF 01 01 13 01,
                01 FF 01 01 13 02, and
                01 FF 01 01 13 03,
```

   for which the corresponding symbolic representations might be:

```
            _GW_impl_Swinging_Squeaky_v1.3_variableA,
            _GW_impl_Swinging_Squeaky_v1.3_variableB, and
            _GW_impl_Swinging_Squeaky_v1.3_variableC.
```

   The names and semantics of implementation-specific variables are not
   otherwise defined by this document, although implementors are
   encouraged to publish such definitions either as appendices to this
   document or by other appropriate means.

   Variable names of which the initial portion is represented
   numerically as 02 and symbolically as "_HOST" are reserved for future
   use. Variable names of which the initial portion is represented
   numerically as 03 and symbolically as "_TS" are similarly reserved.

6. Required Variables

   To the extent that the information represented by a variable defined
   in this section is also represented internally by a gateway for which
   this protocol is realized, access to that variable must be afforded
   by at least one application protocol entity associated with said
   gateway.

6.1. The _GW_version_id Variable

   The variable such that the initial portion of its name is represented
   symbolically as "_GW_version_id" and numerically as:

            01 01 01

   has an octet string value that identifies the protocol entity
   implementation (e.g., "ACME Packet-Whiz Model II").

6.2. The _GW_version_rev Variable

   The variable such that the initial portion of its name is represented
   symbolically as "_GW_version_rev" and numerically as:

            01 01 02

   has an integer value that identifies the revision level of the entity
   implementation. The encoding of the revision level as an integer

   value is implementation-specific.

6.3. The _GW_cfg_nnets Variable

   The variable such that the initial portion of its name is represented
   symbolically as "_GW_cfg_nnets" and numerically as:

                    01 02 01

   has an integer value that represents the number of logical network
   interfaces afforded by the configuration of the gateway.

6.4. Network Interface Variables

   This section describes a related set of variables that represent
   attributes of the logical network interfaces afforded by the gateway
   configuration. Each such network interface is uniquely identified by
   an octet string. The convention by which names are assigned to the
   network interfaces of a gateway is implementation-specific.

6.4.1. The _GW_net_if_type Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_net_if_type" and numerically as:

                    01 03 01 03

   has an integer value that represents the type of the network
   interface identified by the remainder of the name for said variable.
   The value of a variable of this class represents network type
   according to the conventions described in Appendix 1.

6.4.2. The _GW_net_if_speed Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_net_if_speed" and numerically as:

                    01 03 01 04

   has an integer value that represents the estimated nominal bandwidth
   in bits per second of the network interface identified by the
   remainder of the name for said variable.

6.4.3. The _GW_net_if_in_pkts Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_net_if_in_pkts" and numerically as:

            01 03 01 01 01

   has an integer value that represents the number of packets received
   by the gateway over the network interface identified by the remainder
   of the name for said variable.

6.4.4. The _GW_net_if_out_pkts Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_net_if_out_pkts" and numerically as:

            01 03 01 02 01

   has an integer value that represents the number of packets
   transmitted by the gateway over the network interface identified by
   the remainder of the name for said variable.

6.4.5. The _GW_net_if_in_bytes Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_net_if_in_bytes" and numerically as:

            01 03 01 01 02

   has an integer value that represents the number of octets received by
   the gateway over the network interface identified by the remainder of
   the name for said variable.

6.4.6. The _GW_net_if_out_bytes Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_net_if_out_bytes" and numerically as:

            01 03 01 02 02

   has an integer value that represents the number of octets transmitted
   by the gateway over the network interface identified by the remainder
   of the name for said variable.

6.4.7. The _GW_net_if_in_errors Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_net_if_in_errors" and numerically as:

            01 03 01 01 03

   has an integer value that represents the number of reception errors
   encountered by the gateway on the network interface identified by the

Davin, Case, Fedor and Schoffstall   [Page 16]

remainder of the name for said variable. The definition of a
reception error is implementation-specific and may vary according to
network type.

6.4.8. The _GW_net_if_out_errors Variable Class

A variable such that the initial portion of its name is represented
symbolically as "_GW_net_if_out_errors" and numerically as:

         01 03 01 02 03

has an integer value that represents the number of transmission
errors encountered by the gateway on the network interface identified
by the remainder of the name for said variable. The definition of a
transmission error is implementation-specific and may vary according
to network type.

6.4.9. The _GW_net_if_status Variable Class

A variable such that the initial portion of its name is represented
symbolically as "_GW_net_if_status" and numerically as:

         01 03 01 05

has an integer value that represents the current status of the
network interface identified by the remainder of the name for said
variable. Network status is represented according to the conventions
described in Appendix 2.

6.5. Internet Protocol Variables

This section describes variables that represent information related
to protocols and mechanisms of the Internet Protocol (IP) family [3].

6.5.1. Protocol Address Variable Classes

This section describes a related set of variables that represent
attributes of the the IP interfaces presented by a gateway on the
various networks to which it is attached. Each such protocol
interface is uniquely identified by an octet string. The convention
by which names are assigned to the protocol interfaces for a gateway
is implementation-specific.

6.5.1.1. The _GW_pr_in_addr_value Variable Class

A variable such that the initial portion of its name is represented
symbolically as "_GW_pr_in_addr_value" and numerically as:

                01 04 01 01 01

   has an octet string value that literally represents the 32-bit
   Internet address for the IP interface identified by the remainder of
   the name for said variable.

6.5.1.2. The _GW_pr_in_addr_scope Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_pr_in_addr_scope" and numerically as:

                01 04 01 01 02

   has an octet string value that names the network interface with which
   the IP interface identified by the remainder of the name for said
   variable is associated.

6.5.2. Exterior Gateway Protocol (EGP) Variables

   This section describes variables that represent information related
   to protocols and mechanisms of the EGP protocol [4].

6.5.2.1. The _GW_pr_in_egp_core Variable

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_pr_in_egp_core" and numerically as:

                01 04 01 03 01

   has an integer value that characterizes the associated gateway with
   respect to the set of INTERNET core gateways. A nonzero value
   indicates that the associated gateway is part of the INTERNET core.

6.5.2.2. The _GW_pr_in_egp_as Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_pr_in_egp_as" and numerically as:

                01 04 01 03 02

   has an integer value that literally identifies an Autonomous System
   to which this gateway belongs.

6.5.2.3. The EGP Neighbor Variable Classes

   This section describes a related set of variables that represent
   attributes of "neighbors" with which the gateway may be associated by
   EGP. Each such EGP neighbor is uniquely identified by an octet

string. The convention by which names are assigned to EGP neighbors
of a gateway is implementation-specific.

6.5.2.3.1. The _GW_pr_in_egp_neighbor_addr Variable Class

A variable such that the initial portion of its name is represented
symbolically as "_GW_pr_in_egp_neighbor_addr" and numerically as:

           01 04 01 03 03 01

has an octet string value that literally represents the 32-bit
Internet address for the EGP neighbor identified by the remainder of
the name for said variable.

6.5.2.3.2. The _GW_pr_in_egp_neighbor_state Variable Class

A variable such that the initial portion of its name is represented
symbolically as "_GW_pr_in_egp_neighbor_state" and numerically as:

           01 04 01 03 03 02

has an octet string value that represents the EGP protocol state of
the gateway with respect to the EGP neighbor identified by the
remainder of the name for said variable. The meaningful values for
such a variable are: "IDLE," "ACQUISITION," "DOWN," "UP," and
"CEASE."

6.5.2.4. The _GW_pr_in_egp_errors Variable

The variable such that the initial portion of its name is represented
symbolically as "_GW_pr_in_egp_errors" and numerically as:

           01 04 01 03 05

has an integer value that represents the number of EGP protocol
errors.

6.5.3. Routing Variable Classes

This section describes a related set of variables that represent
attributes of the the IP routes by which a gateway directs packets to
various destinations on the Internet. Each such route is uniquely
identified by an octet string that is the concatenation of the
literal 32-bit value of the Internet address for the destination of
said route together with an implementation-specific octet string.
The convention by which names are assigned to the Internet routes for
a gateway is in all other respects implementation-specific.

6.5.3.1. The _GW_pr_in_rt_gateway Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_pr_in_rt_gateway" and numerically as:

                 01 04 01 02 01

   has an octet string value that literally represents the 32-bit
   Internet address of the next gateway to which traffic is directed by
   the route identified by the remainder of the name for said variable.

6.5.3.2. The _GW_pr_in_rt_type Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_pr_in_rt_type" and numerically as:

                 01 04 01 02 02

   has an integer value that represents the type of the route identified
   by the remainder of the name for said variable. Route types are
   identified according to the conventions described in Appendix 3.

6.5.3.3. The _GW_pr_in_rt_how-learned Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_pr_in_rt_how-learned" and numerically as:

                 01 04 01 02 03

   has an octet string value that represents the source of the
   information from which the route identified by the remainder of the
   name for said variable is generated. The meaningful values of such a
   variable are: "STATIC," "EGP," and "RIP."

6.5.3.4. The _GW_pr_in_rt_metric0 Variable Class

   A variable such that the initial portion of its name is represented
   symbolically as "_GW_pr_in_rt_metric0" and numerically as:

                 01 04 01 02 04

   has an integer value that represents the quality (in terms of cost,
   distance from the ultimate destination, or other metric) of the route
   identified by the remainder of the name for said variable.

6.5.3.5. The _GW_pr_in_rt_metric1 Variable Class

   A variable such that the initial portion of its name is represented

symbolically as "_GW_pr_in_rt_metric1" and numerically as:

                    01 04 01 02 05

has an integer value that represents the quality (in terms of cost, distance from the ultimate destination, or other metric) of the route identified by the remainder of the name for said variable.

6.6. DECnet Protocol Variables

This section describes variables that represent information related to protocols and mechanisms of the DEC Digital Network Architecture. DEC and DECnet are registered trademarks of Digital Equipment Corporation.

6.7. XNS Protocol Variables

This section describes variables that represent information related to protocols and mechanisms of the Xerox Network System. Xerox Network System and XNS are registered trademarks of the XEROX Corporation.

7. Implementation-Specific Variables

Additional variables that may be presented for inspection or manipulation by particular protocol entity implementations are described in Appendices to this document.

8. References

    [1]   CCITT, "Message Handling Systems: Presentation Transfer
          Syntax and Notation", Recommendation X.409, 1984.

    [2]   Postel, J., "User Datagram Protocol", RFC-768,
          USC/Information Sciences Institute, August 1980.

    [3]   Postel, J., "Internet Protocol", RFC-760, USC/Information
          Sciences Institute, January 1980.

    [4]   Rosen, E., "Exterior Gateway Protocol", RFC-827, Bolt
          Beranek and Newman, October 1982.

9. Appendix 1: Network Type Representation

Numeric representations for various types of networks are presented below:

```
                    Value    Network Type
                    ====================
                    0        Unspecified
                    1        IEEE 802.3 MAC
                    2        IEEE 802.4 MAC
                    3        IEEE 802.5 MAC
                    4        Ethernet
                    5        ProNET-80
                    6        ProNET-10
                    7        FDDI
                    8        X.25
                    9        Point-to-Point Serial
                    10       Proprietary Point-to-Point Serial
                    11       ARPA 1822 HDH
                    12       ARPA 1822
                    13       AppleTalk
                    14       StarLAN
```

10. Appendix 2: Network Status Representation

Numeric representations for network status are presented below.

```
                    Value    Network Status
                    ======================
                    0        Interface Operating Normally
                    1        Interface Not Present
                    2        Interface Disabled
                    3        Interface Down
                    4        Interface Attempting Link
```

11. Appendix 3: Route Type Representation

Numeric representations for route types are presented below.

```
                    Value    Route Type
                    ==================
                    0        Route to Nowhere -- ignored
                    1        Route to Directly Connected Network
                    2        Route to a Remote Host
                    3        Route to a Remote Network
                    4        Route to a Sub-Network
```

12. Appendix 4: Initial Implementation Strategy

   The initial objective of implementing the protocol specified in this
   document is to provide a mechanism for monitoring Internet gateways.
   While the protocol design makes some provision for gateway management

Davin, Case, Fedor and Schoffstall   [Page 22]

functions as well, this aspect of the design is not fully developed
and needs further refinement before a generally useful implementation
could be produced. Accordingly, initial implementations will not
generate or respond to the optional Set Request message type.

The protocol defined here may be subsequently refined based upon
experience with early implementations or upon further study of the
problem of gateway management. Moreover, it may be superceded by
other proposals in the area of gateway monitoring and control.

Implementations of the authentication protocol specified in this
document are likely to evolve in response to the particular security
and privacy needs of its users. While, in general, the association
between particular half-sessions of the authentication protocol and
the described triplets of functions is specific to an implementation
and beyond the scope of this document, the desire for immediate
interoperability among initial implementations of this protocol is
best served by the temporary adoption of a common authentication
scheme. Accordingly, initial implementations will associate with
every possible half-session a triplet of functions that realizes a
trivial authentication mechanism:

(1) The authentication function is defined to have the value
     TRUE over the entire domain of authentication protocol
     messages.

(2) The message interpretation function is defined to be the
     identity function.

(3) The message representation function is defined to be the
     identity function.

Because this initial posture with respect to authentication is not
likely to remain acceptable indefinitely, implementors are urged to
adopt designs that isolate authentication mechanism as much as
possible from other components of the implementation.

13. Appendix 5: Routing Information Propagation Variables

This section describes a set of related variables that characterize
the sources and destinations of routing information propagated by
various routing protocols. These variables have meaning only for
those routing protocol implementations that afford greater
flexibility in propagating routing information than is required by
the various routing protocol specifications.

Each IP interface afforded by the configuration of the gateway over
which routing information may propagate via a routing protocol

(target interface) is named by a string of four octets that literally
represents the IP address associated with said protocol interface.

Each IP protocol interface afforded by the configuration of the
gateway over which routing information may arrive via any routing
protocol (source interface) is named by a string of four octets that
literally represents the IP address associated with said protocol
interface.

Each routing protocol by which a gateway receives information that it
uses to route IP traffic (source routing protocol) is named by a
single-octet string according to the conventions set forth in
Appendix 6 of this document.

Each routing protocol by which a gateway propagates routing
information used by other hosts or gateways to route IP traffic
(target routing protocol) is named by a single-octet string according
to the conventions set forth in Appendix 6 of this document.

A variable such that the initial portion of its name is the
concatenation of:

(1)   the octet string represented symbolically as "_GW_pr_in_rif"
      and numerically as 01 04 01 04 followed by:

(2)   the name of a target routing protocol followed by

(3)   the name of a target interface followed by

(4)   the name of a source routing protocol followed by

(5)   the name of a source interface

has an integer value that characterizes the propagation of routing
information between the sources and destinations of such information
that are identified by the initial portion of that variable's name. A
non-zero value for such a variable indicates that routing information
received via the source routing protocol named by the fourth
component of the variable name on the source interface named by its
fifth component is propagated via the target routing protocol named
by the second component of the variable name over the target
interface named by its third component. A zero value for such a
variable indicates that routing information received via the source
routing protocol on the source interface identified in the variable
name is NOT propagated via the target routing protocol over the
target interface identified in the variable name.

14. Appendix 6: Routing Protocol Representation

Numeric representations for routing protocols are presented below.

Value	Routing Protocol
0	None -- Reserved
1	Berkeley RIP Version 1
2	EGP
3	GGP
4	Hello
5	Other IGRP

15. Appendix 7: Proteon p4200 Release 7.4 Variables

This section describes implementation-specific variables presented by
the implementation of this protocol in Software Release 7.4 for the
Proteon p4200 Internet Router. These variable definitions are
subject to change without notice.

15.1. The Network Interface Variables

This section describes a related set of variables that represent
attributes of a network interface in the Proteon p4200 Internet
Router gateway. Each such network interface is uniquely named by an
implementation-specific octet string of length 1.

15.1.1. The Generic Network Interface Variables

This section describes a related set of variables that represent
attributes common to all network interfaces in the Proteon p4200
Internet Router gateway. Each generic network interface of a p4200
configuration is uniquely named by the concatenation of the octet
string represented symbolically as "_GW_impl_Proteon_p4200-R7.4_net-
if" and numerically as:

        01 FF 01 01 01

followed by the name of said network interface as described above.

15.1.1.1. The Generic _ovfl-in Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a generic network interface followed by
the octet string represented symbolically as "_ovfl-in" and
numerically as 01, has an integer value that represents the number of
input packets dropped due to gateway congestion for the network
interface identified by the initial portion of its name.

Davin, Case, Fedor and Schoffstall   [Page 25]

15.1.1.2. The Generic _ovfl-out Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a generic network interface followed by
   the octet string represented symbolically as "_ovfl-out" and
   numerically as 02, has an integer value that represents the number of
   output packets dropped due to gateway congestion for the network
   interface identified by the initial portion of its name.

15.1.1.3. The Generic _slftst-pass Variable Class   A variable
   such that the initial portion of its name is the concatenation of the
   name for a generic network interface followed by the octet string
   represented symbolically as "_slftst-pass" and numerically as 03, has
   an integer value that represents the number of times the interface
   self-test procedure succeeded for the network interface identified by
   the initial portion of its name.
15.1.1.4. The Generic _slftst-fail Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a generic network interface followed by
   the octet string represented symbolically as "_slftst-fail" and
   numerically as 04, has an integer value that represents the number of
   times the interface self-test procedure failed for the network
   interface identified by the initial portion of its name.

15.1.1.5. The Generic _maint-fail Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a generic network interface followed by
   the octet string represented symbolically as "_maint-fail" and
   numerically as 06, has an integer value that represents the number of
   times the network maintenance procedure failed for the network
   interface identified by the initial portion of its name.

15.1.1.6. The Generic _csr Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a generic network interface followed by
   the octet string represented symbolically as "_csr" and numerically
   as 07, has an integer value that represents the internal address of
   the device CSR for the network interface identified by the initial
   portion of its name.

15.1.1.7. The Generic _vec Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a generic network interface followed by
   the octet string represented symbolically as "_vec" and numerically

as 08, has an integer value that identifies the device interrupt
vector used by the network interface identified by the initial
portion of its name.

15.1.2. The ProNET Network Interface Variables

This section describes a related set of variables that represent
attributes of a ProNET type network interface in the Proteon p4200
Internet Router gateway. Each network interface of a p4200
configuration that supports ProNET media is uniquely named by the
concatenation of the octet string represented symbolically as
"_GW_impl_Proteon_p4200-R7.4_devpn" and numerically as:

            01 FF 01 01 04

followed by the name of said network interface as described above.

15.1.2.1. The ProNET _node-number Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a ProNET type network interface
followed by the octet string represented symbolically as "_node-
number" and numerically as 01, has an integer value that represents
the ProNET node number associated with the network interface
identified by the initial portion of its name.

15.1.2.2. The ProNET _in-data-present Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a ProNET type network interface
followed by the octet string represented symbolically as "_in-data-
present" and numerically as 02, has an integer value that represents
the number of times that unread data was present in the input packet
buffer for the network interface dentified by the initial portion of
its name.

15.1.2.3. The ProNET _in-overrun Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a ProNET type network interface
followed by the octet string represented symbolically as "_in-
overrun" and numerically as 03, has an integer value that represents
the number of times that a packet copied from the ring exceeded the
size of the packet input buffer on the network interface identified
by the initial portion of its name.

15.1.2.4.   The ProNET _in-odd-byte-cnt Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a ProNET type network interface
   followed by the octet string represented symbolically as "_in-odd-
   byte-cnt" and numerically as 04, has an integer value that represents
   the number of times that a packet was received with an odd number of
   bytes on the network interface identified by the initial portion of
   its name.

15.1.2.5.   The ProNET _in-parity-error Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a ProNET type network interface
   followed by the octet string represented symbolically as "_in-
   parity-error" and numerically as 05, has an integer value that
   represents the number of times that a parity error was detected in a
   packet copied from the ring on the network interface identified by
   the initial portion of its name.

15.1.2.6.   The ProNET _in-bad-format Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a ProNET type network interface
   followed by the octet string represented symbolically as "_in-bad-
   format" and numerically as 06, has an integer value that represents
   the number of times that a format error was detected in a packet
   copied from the ring on the network interface identified by the
   initial portion of its name.

15.1.2.7.   The ProNET _not-in-ring Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a ProNET type network interface
   followed by the octet string represented symbolically as "_not-in-
   ring" and numerically as 07, has an integer value that represents the
   number of times that the ProNET wire center relays were detected in
   an unenergized state for the network interface identified by the
   initial portion of its name.

15.1.2.8.   The ProNET _out-ring-inits Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a ProNET type network interface
   followed by the octet string represented symbolically as "_out-ring-
   inits" and numerically as 08, has an integer value that represents
   the number of times that ring initialization has been attempted on
   the network interface identified by the initial portion of its name.

15.1.2.9.   The ProNET _out-bad-format Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a ProNET type network interface
followed by the octet string represented symbolically as "_out-bad-
format" and numerically as 09, has an integer value that represents
the number of times that an improperly formatted packet was detected
in the course of an output operation on the network interface
identified by the initial portion of its name.

15.1.2.10.   The ProNET _out-timeout Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a ProNET type network interface
followed by the octet string represented symbolically as "_out-
timeout" and numerically as 0A, has an integer value that represents
the number of times that an attempt to originate a message has been
delayed by more than 700 ms on the network interface identified by
the initial portion of its name.

15.1.3.   The Ethernet Network Interface Variables

This section describes a related set of variables that represent
attributes of an Ethernet type network interface in the Proteon p4200
Internet Router gateway. Each network interface of a p4200
configuration that supports Ethernet media is uniquely named by the
concatenation of the octet string represented symbolically as
"_GW_impl_Proteon_p4200-R7.4_dev-ie" and numerically as:

                01 FF 01 01 03

followed by the name of said network interface as described above.

15.1.3.1.   The Ethernet _phys-addr Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_phys-addr"
and numerically as 01 has an octet string value that literally
represents the Ethernet station address associated with the network
interface identified by the initial portion of its name.

15.1.3.2.   The Ethernet _input-ovfl Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_input-
ovfl" and numerically as 02, has an integer value that represents the

number of times the size of a received frame exceeded the maximum
allowable for the network interface identified by the initial portion
of its name.

15.1.3.3. The Ethernet _input-dropped Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented0 symbolically as "_input-
dropped" and numerically as 03, has an integer value that represents
the number of times the loss of one or more frames was detected on
the network interface identified by the initial portion of its name.

15.1.3.4. The Ethernet _output-retry Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_output-
retry" and numerically as 04, has an integer value that represents
the number of output operations retried as the result of a
transmission failure on the network interface identified by the
initial portion of its name.

15.1.3.5. The Ethernet _output-fail Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_output-
fail" and numerically as 05, has an integer value that represents the
number of failed output operations detected on the network interface
identified by the initial portion of its name.

15.1.3.6. The Ethernet _excess-coll Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_excess-
coll" and numerically as 06, has an integer value that represents the
number of times a transmit frame incurred 16 successive collisions
when attempting media access via the network interface identified by
the initial portion of its name.

15.1.3.7. The Ethernet _frag-rcvd Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_frag-rcvd"
and numerically as 07, has an integer value that represents the

number of collision fragments (i.e., "runt packets") that were
received and filtered by the controller for the network interface
identified by the initial portion of its name.

15.1.3.8.   The Ethernet _frames-lost Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_frames-
lost" and numerically as 08, has an integer value that represents the
number of frames not accepted by the Receive FIFO due to insufficient
space for the network interface identified by the initial portion of
its name.

15.1.3.9.   The Ethernet _multicst-accept Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_multicst-
accept" and numerically as 09, has an integer value that represents
the number of frames received with a multicast-group destination
address that matches one of those assigned to the controller for the
network interface identified by the initial portion of said variable
name.

15.1.3.10.   The Ethernet _multicst-reject Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_multicst-
reject" and numerically as 0A, has an integer value that represents
the number of frames detected as having a multicast-group destination
address that matches none of those assigned to the controller for the
network interface identified by the initial portion of said variable
name.

15.1.3.11.   The Ethernet _crc-error Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_crc-error"
and numerically as 0B, has an integer value that represents the
number of frames received with a CRC error on the network interface
identified by the initial portion of its name.

15.1.3.12.   The Ethernet _alignmnt-error Variable Class

A variable such that the initial portion of its name is the

concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_alignmnt-
error" and numerically as 0C, has an integer value that represents
the number of frames received with an alignment error on the network
interface identified by the initial portion of its name. A received
frame is said to have an alignment error if its received length is
not an integral multiple of 8 bits.

15.1.3.13. The Ethernet _collisions Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as
"_collisions" and numerically as 0D, has an integer value that
represents the number of collisions incurred during transmissions on
the network interface identified by the initial portion of its name.

15.1.3.14. The Ethernet _out-of-window-coll Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for an Ethernet type network interface
followed by the octet string represented symbolically as "_out-of-
window-coll" and numerically as 0E, has an integer value that
represents the number of out-ofwindow collisions incurred during
transmissions on the network interface identified by the initial
portion of its name. Outof-window collisions are those occurring
after the first 51.2 microseconds of slot time.

15.1.4. The Serial Network Interface Variables

This section describes a related set of variables that represent
attributes of an serial line type network interface in the Proteon
p4200 Internet Router gateway. Each network interface of a p4200
configuration that supports serial communications is uniquely named
by the concatenation of the octet string represented symbolically as
"_GW_impl_Proteon_p4200-R7.4_dev-sl" and numerically as:

        01 FF 01 01 05

followed by the name of said network interface as described above.

15.1.4.1. The Serial _tx-pkts Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_tx-pkts"
and numerically as 01, has an integer value that represents the
number of packets transmitted on the network interface identified by

the initial portion of its name.

15.1.4.2. The Serial _tx-framing-error Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_tx-
framing-error" and numerically as 02, has an integer value that
represents the number of transmission framing errors for the network
interface identified by the initial portion of its name.

15.1.4.3. The Serial _tx-underrns Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_tx-
underrns" and numerically as 03, has an integer value that represents
the number of transmission underrun errors for the network interface
identified by the initial portion of its name.

15.1.4.4. The Serial _tx-no-dcd Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_tx-no-dcd"
and numerically as 04, has an integer value that represents the
number of times transmission failed due to absence of the EIA Data
Carrier Detect signal on the network interface identified by the
initial portion of its name.

15.1.4.5. The Serial _tx-no-cts Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_tx-no-cts"
and numerically as 05, has an integer value that represents the
number of times transmission failed due to absence of the EIA Clear
To Send signal on the network interface identified by the initial
portion of its name.

15.1.4.6. The Serial _tx-no-dsr Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_tx-no-dsr"
and numerically as 06, has an integer value that represents the
number of times transmission failed due to absence of the EIA Data
Set Ready signal on the network interface identified by the initial

portion of its name.

15.1.4.7. The Serial _rx-pkts Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_rx-pkts"
and numerically as 07, has an integer value that represents the
number of packets received on the network interface identified by the
initial portion of its name.

15.1.4.8. The Serial _rx-framing-err Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_rx-
framing-err" and numerically as 08, has an integer value that
represents the number of receive framing errors on the network
interface identified by the initial portion of its name.

15.1.4.9. The Serial _rx-overrns Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_rx-
overrns" and numerically as 09, has an integer value that represents
the number of receive overrun errors on the network interface
identified by the initial portion of its name.

15.1.4.10. The Serial _rx-aborts Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_rx-aborts"
and numerically as 0A, has an integer value that represents the
number of aborted frames received on the network interface identified
by the initial portion of its name.

15.1.4.11. The Serial _rx-crc-err Variable Class

A variable such that the initial portion of its name is the
concatenation of the name for a serial line type network interface
followed by the octet string represented symbolically as "_rx-crc-
err" and numerically as 0B, has an integer value that represents the
number of frames received with CRC errors on the network interface
identified by the initial portion of its name.

15.1.4.12. The Serial _rx-buf-ovfl Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a serial line type network interface
   followed by the octet string represented symbolically as "_rx-buf-
   ovfl" and numerically as 0C, has an integer value that represents the
   number of received frames whose size exceeded the maximum allowable
   on the network interface identified by the initial portion of its
   name.

15.1.4.13. The Serial _rx-buf-locked Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a serial line type network interface
   followed by the octet string represented symbolically as "_rx-buf-
   locked" and numerically as 0D, has an integer value that represents
   the number of received frames lost for lack of an available buffer on
   the network interface identified by the initial portion of its name.

15.1.4.14. The Serial _rx-line-speed Variable Class

   A variable such that the initial portion of its name is the
   concatenation of the name for a serial line type network interface
   followed by the octet string represented symbolically as "_rx-line-
   speed" and numerically as 0E, has an integer value that represents an
   estimate of serial line bandwidth in bits per second for the network
   interface identified by the initial portion of its name.