SNMPv2-TC DEFINITIONS ::= BEGIN

IMPORTS
   IpAddress
   FROM SNMPv2-SMI;

-- Textual conventions

DisplayString ::= TEXTUAL-CONVENTION
   DISPLAY-HINT
      "255a"
   STATUS current
   DESCRIPTION
      "Represents textual information taken from the NVT ASCII
      character set, as defined in pages 4, 10-11 of RFC 854.

To summarize RFC 854, the NVT ASCII repertoire specifies:

- the use of character codes 0-127 (decimal)
- the graphics characters (32-126) are interpreted as
  US ASCII
- NUL, LF, CR, BEL, BS, HT, VT and FF have the special
  meanings specified in RFC 854
- the other 25 codes have no standard interpretation
- the sequence 'CR LF' means newline
- the sequence 'CR NUL' means carriage-return
- an 'LF' not preceded by a 'CR' means moving to the
  same column on the next line.
- the sequence 'CR x' for any x other than LF or NUL is
  illegal. (Note that this also means that a string may
  end with either 'CR LF' or 'CR NUL', but not with CR.)

Any object defined using this syntax may not exceed 255
characters in length."
   SYNTAX OCTET STRING (SIZE (0..255))

PhysAddress ::= TEXTUAL-CONVENTION
   DISPLAY-HINT
      "1x:"
   STATUS current
   DESCRIPTION
      "Represents media- or physical-level addresses."
   SYNTAX OCTET STRING

MacAddress ::= TEXTUAL-CONVENTION
   DISPLAY-HINT
      "1x:"
   STATUS current
   DESCRIPTION
      "Represents an 802 MAC address represented in the
      'canonical' order defined by IEEE 802.1a, i.e., as if it
      were transmitted least significant bit first, even though
      802.5 (in contrast to other 802.x protocols) requires MAC
      addresses to be transmitted most significant bit first."
   SYNTAX OCTET STRING (SIZE (6))

TruthValue ::= TEXTUAL-CONVENTION
   STATUS current
TestAndIncr ::= TEXTUAL-CONVENTION
  STATUS current
  DESCRIPTION
  "Represents integer-valued information used for atomic operations. When the management protocol is used to specify that an object instance having this syntax is to be modified, the new value supplied via the management protocol must precisely match the value presently held by the instance. If not, the management protocol set operation fails with an error of 'inconsistentValue'. Otherwise, if the current value is the maximum value of $2^{31}-1$ (2147483647 decimal), then the value held by the instance is wrapped to zero; otherwise, the value held by the instance is incremented by one. (Note that regardless of whether the management protocol set operation succeeds, the variable-binding in the request and response PDUs are identical.)

The value of the ACCESS clause for objects having this syntax is either 'read-write' or 'read-create'. When an instance of a columnar object having this syntax is created, any value may be supplied via the management protocol.

When the network management portion of the system is re-initialized, the value of every object instance having this syntax must either be incremented from its value prior to the re-initialization, or (if the value prior to the re-initialization is unknown) be set to a pseudo-randomly generated value."

SYNTAX INTEGER (0..2147483647)

AutonomousType ::= TEXTUAL-CONVENTION
  STATUS current
  DESCRIPTION
  "Represents an independently extensible type identification value. It may, for example, indicate a particular sub-tree with further MIB definitions, or define a particular type of protocol or hardware."

SYNTAX OBJECT IDENTIFIER

InstancePointer ::= TEXTUAL-CONVENTION
  STATUS obsolete
  DESCRIPTION
  "A pointer to either a specific instance of a MIB object or a conceptual row of a MIB table in the managed device. In the latter case, by convention, it is the name of the particular instance of the first accessible columnar object in the conceptual row.

The two uses of this textual convention are replaced by VariablePointer and RowPointer, respectively."

SYNTAX OBJECT IDENTIFIER

VariablePointer ::= TEXTUAL-CONVENTION
  STATUS current
  DESCRIPTION
  "A pointer to a specific object instance. For example, sysContact.0 or ifInOctets.3."

SYNTAX OBJECT IDENTIFIER

RowPointer ::= TEXTUAL-CONVENTION
  STATUS current
  DESCRIPTION
  "Represents a pointer to a conceptual row. The value is the name of the instance of the first accessible columnar object in the conceptual row.

For example, ifIndex.3 would point to the 3rd row in the ifTable (note that if ifIndex were not-accessible, then
**SYNTAX OBJECT IDENTIFIER**

RowStatus ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The RowStatus textual convention is used to manage the creation and deletion of conceptual rows, and is used as the value of the SYNTAX clause for the status column of a conceptual row (as described in Section 7.7.1 of [2].)

The status column has six defined values:

- 'active', which indicates that the conceptual row is available for use by the managed device;

- 'notInService', which indicates that the conceptual row exists in the agent, but is unavailable for use by the managed device (see NOTE below);

- 'notReady', which indicates that the conceptual row exists in the agent, but is missing information necessary in order to be available for use by the managed device;

- 'createAndGo', which is supplied by a management station wishing to create a new instance of a conceptual row and to have its status automatically set to active, making it available for use by the managed device;

- 'createAndWait', which is supplied by a management station wishing to create a new instance of a conceptual row (but not make it available for use by the managed device); and,

- 'destroy', which is supplied by a management station wishing to delete all of the instances associated with an existing conceptual row.

Whereas five of the six values (all except 'notReady') may be specified in a management protocol set operation, only three values will be returned in response to a management protocol retrieval operation: 'notReady', 'notInService' or 'active'. That is, when queried, an existing conceptual row has only three states: it is either available for use by the managed device (the status column has value 'active'); it is not available for use by the managed device, though the agent has sufficient information to make it so (the status column has value 'notInService'); or, it is not available for use by the managed device, and an attempt to make it so would fail because the agent has insufficient information (the state column has value 'notReady').

**NOTE WELL**

This textual convention may be used for a MIB table, irrespective of whether the values of that table's conceptual rows are able to be modified while it is active, or whether its conceptual rows must be taken out of service in order to be modified. That is, it is the responsibility of the DESCRIPTION clause of the status column to specify whether the status column must not be 'active' in order for the value of some other column of the same conceptual row to be modified. If such a specification is made, affected columns may be changed by an SNMP set PDU if the RowStatus would not be equal to 'active' either immediately before or after processing the PDU. In other words, if the PDU also contained a varbind that would change the RowStatus value, the column in question may be changed if the RowStatus was not equal to 'active' as the PDU was received, or if the varbind sets the status to a value other than 'active'.

ifDescr.3 would be used instead)."
Also note that whenever any elements of a row exist, the RowStatus column must also exist.

To summarize the effect of having a conceptual row with a status column having a SYNTAX clause value of RowStatus, consider the following state diagram:

```
<table>
<thead>
<tr>
<th>ACTION</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>set status</td>
<td>status column does not exist</td>
<td>status column is notReady</td>
<td>status column is notInService</td>
<td>status column is active</td>
</tr>
<tr>
<td>column to</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>createAndGo</td>
<td>noError or/ or inconsistently</td>
<td>inconsistently</td>
<td>inconsistently</td>
<td>inconsistently</td>
</tr>
<tr>
<td></td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
<td>Value</td>
</tr>
<tr>
<td>set status</td>
<td>noError see 1 or wrongValue</td>
<td>inconsistently</td>
<td>inconsistently</td>
<td>inconsistently</td>
</tr>
<tr>
<td>column to</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>createAndWait</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set status</td>
<td>inconsistently</td>
<td>inconsistently</td>
<td>noError</td>
<td>noError</td>
</tr>
<tr>
<td>column to</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>active</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set status</td>
<td>inconsistently</td>
<td>inconsistently</td>
<td>noError</td>
<td>noError</td>
</tr>
<tr>
<td>column to</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>notInService</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set status</td>
<td>noError</td>
<td>noError</td>
<td>noError</td>
<td>noError</td>
</tr>
<tr>
<td>column to destroy</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>set any other</td>
<td>see 4</td>
<td>noError</td>
<td>noError</td>
<td>see 5</td>
</tr>
<tr>
<td>column to some</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1) goto B or C,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>depending on</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>information</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>available to the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>agent.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2) if other</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>variable bindings</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>included in the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>same PDU, provide</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>values for all</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>columns which are</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>missing but</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>required, then</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>return noError and</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>goto D.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(3) if other</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>variable bindings</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>included in the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>same PDU, provide</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>values for all</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>columns which are</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>missing but</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>required, then</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>return noError and</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>goto C.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(4) at the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>discretion of the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>agent, the return</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value may be</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>either:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inconsistentName:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>because the agent</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>does not choose</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to create such an</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>instance when the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>corresponding</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RowStatus instance</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>does not exist, or</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inconsistentValue:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>if the supplied</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value is</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>inconsistent with</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the state of some</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>other MIB object's</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>value, or</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>noError: because</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>the agent chooses</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>to create the</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>instance.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If noError is returned, then the instance of the status column must also be created, and the new state is B or C, depending on the information available to the agent. If inconsistentName or inconsistentValue is returned, the row
remains in state A.

(5) depending on the MIB definition for the column/table, either noError or inconsistentValue may be returned.

NOTE: Other processing of the set request may result in a response other than noError being returned, e.g., wrongValue, noCreation, etc.

Conceptual Row Creation

There are four potential interactions when creating a conceptual row: selecting an instance-identifier which is not in use; creating the conceptual row; initializing any objects for which the agent does not supply a default; and, making the conceptual row available for use by the managed device.

Interaction 1: Selecting an Instance-Identifier

The algorithm used to select an instance-identifier varies for each conceptual row. In some cases, the instance-identifier is semantically significant, e.g., the destination address of a route, and a management station selects the instance-identifier according to the semantics.

In other cases, the instance-identifier is used solely to distinguish conceptual rows, and a management station without specific knowledge of the conceptual row might examine the instances present in order to determine an unused instance-identifier. (This approach may be used, but it is often highly sub-optimal; however, it is also a questionable practice for a naive management station to attempt conceptual row creation.)

Alternately, the MIB module which defines the conceptual row might provide one or more objects which provide assistance in determining an unused instance-identifier. For example, if the conceptual row is indexed by an integer-value, then an object having an integer-valued SYNTAX clause might be defined for such a purpose, allowing a management station to issue a management protocol retrieval operation. In order to avoid unnecessary collisions between competing management stations, 'adjacent' retrievals of this object should be different.

Finally, the management station could select a pseudo-random number to use as the index. In the event that this index was already in use and an inconsistentValue was returned in response to the management protocol set operation, the management station should simply select a new pseudo-random number and retry the operation.

A MIB designer should choose between the two latter algorithms based on the size of the table (and therefore the efficiency of each algorithm). For tables in which a large number of entries are expected, it is recommended that a MIB object be defined that returns an acceptable index for creation. For tables with small numbers of entries, it is recommended that the latter pseudo-random index mechanism be used.

Interaction 2: Creating the Conceptual Row

Once an unused instance-identifier has been selected, the management station determines if it wishes to create and activate the conceptual row in one transaction or in a negotiated set of interactions.

Interaction 2a: Creating and Activating the Conceptual Row

The management station must first determine the column requirements, i.e., it must determine those columns for which it must or must not provide values. Depending on the
complexity of the table and the management station's knowledge of the agent's capabilities, this determination can be locally by the management station. Alternately, the management station issues a management protocol get operation to examine all columns in the conceptual row that it wishes to create. In response, for each column, there are three possible outcomes:

- a value is returned, indicating that some other management station has already created this conceptual row. We return to interaction 1.

- the exception 'noSuchInstance' is returned, indicating that the agent implements the object-type associated with this column, and that this column in at least one conceptual row would be accessible in the MIB view used by the retrieval were it to exist. For those columns to which the agent provides read-create access, the 'noSuchInstance' exception tells the management station that it should supply a value for this column when the conceptual row is to be created.

- the exception 'noSuchObject' is returned, indicating that the agent does not implement the object-type associated with this column or that there is no conceptual row for which this column would be accessible in the MIB view used by the retrieval. As such, the management station can not issue any management protocol set operations to create an instance of this column.

Once the column requirements have been determined, a management protocol set operation is accordingly issued. This operation also sets the new instance of the status column to 'createAndGo'.

When the agent processes the set operation, it verifies that it has sufficient information to make the conceptual row available for use by the managed device. The information available to the agent is provided by two sources: the management protocol set operation which creates the conceptual row, and, implementation-specific defaults supplied by the agent (note that an agent must provide implementation-specific defaults for at least those objects which it implements as read-only). If there is sufficient information available, then the conceptual row is created, a 'noError' response is returned, the status column is set to 'active', and no further interactions are necessary (i.e., interactions 3 and 4 are skipped). If there is insufficient information, then the conceptual row is not created, and the set operation fails with an error of 'inconsistentValue'. On this error, the management station can issue a management protocol retrieval operation to determine if this was because it failed to specify a value for a required column, or, because the selected instance of the status column already existed. In the latter case, we return to interaction 1. In the former case, the management station can re-issue the set operation with the additional information, or begin interaction 2 again using 'createAndWait' in order to negotiate creation of the conceptual row.

**NOTE WELL**

Regardless of the method used to determine the column requirements, it is possible that the management station might deem a column necessary when, in fact, the agent will not allow that particular columnar instance to be created or written. In this case, the management protocol set operation will fail with an error such as 'noCreation' or 'notWritable'. In this case, the management station decides whether it needs to be able to set a value for that particular columnar instance. If not, the management station re-issues the management protocol set operation, but without setting a value for that particular columnar instance;
otherwise, the management station aborts the row creation algorithm.

Interaction 2b: Negotiating the Creation of the Conceptual Row

The management station issues a management protocol set operation which sets the desired instance of the status column to 'createAndWait'. If the agent is unwilling to process a request of this sort, the set operation fails with an error of 'wrongValue'. (As a consequence, such an agent must be prepared to accept a single management protocol set operation, i.e., interaction 2a above, containing all of the columns indicated by its column requirements.) Otherwise, the conceptual row is created, a 'noError' response is returned, and the status column is immediately set to either 'notInService' or 'notReady', depending on whether it has sufficient information to make the conceptual row available for use by the managed device. If there is sufficient information available, then the status column is set to 'notInService'; otherwise, if there is insufficient information, then the status column is set to 'notReady'. Regardless, we proceed to interaction 3.

Interaction 3: Initializing non-defaulted Objects

The management station must now determine the column requirements. It issues a management protocol get operation to examine all columns in the created conceptual row. In the response, for each column, there are three possible outcomes:

- a value is returned, indicating that the agent implements the object-type associated with this column and has sufficient information to provide a value. For those columns to which the agent provides read-create access (and for which the agent allows their values to be changed after their creation), a value return tells the management station that it may issue additional management protocol set operations, if it desires, in order to change the value associated with this column.

- the exception 'noSuchInstance' is returned, indicating that the agent implements the object-type associated with this column and that this column in at least one conceptual row would be accessible in the MIB view used by the retrieval were it to exist. However, the agent does not have sufficient information to provide a value, and until a value is provided, the conceptual row may not be made available for use by the managed device. For those columns to which the agent provides read-create access, the 'noSuchInstance' exception tells the management station that it must issue additional management protocol set operations, in order to provide a value associated with this column.

- the exception 'noSuchObject' is returned, indicating that the agent does not implement the object-type associated with this column or that there is no conceptual row for which this column would be accessible in the MIB view used by the retrieval. As such, the management station can not issue any management protocol set operations to create an instance of this column.

If the value associated with the status column is 'notReady', then the management station must first deal with all 'noSuchInstance' columns, if any. Having done so, the value of the status column becomes 'notInService', and we proceed to interaction 4.

Interaction 4: Making the Conceptual Row Available

Once the management station is satisfied with the values associated with the columns of the conceptual row, it issues a management protocol set operation to set the status column
to `active'. If the agent has sufficient information to make the conceptual row available for use by the managed device, the management protocol set operation succeeds (a `noError' response is returned). Otherwise, the management protocol set operation fails with an error of `inconsistentValue'.

NOTE WELL
A conceptual row having a status column with value `notInService' or `notReady' is unavailable to the managed device. As such, it is possible for the managed device to create its own instances during the time between the management protocol set operation which sets the status column to `createAndWait' and the management protocol set operation which sets the status column to `active'. In this case, when the management protocol set operation is issued to set the status column to `active', the values held in the agent supersede those used by the managed device.

If the management station is prevented from setting the status column to `active' (e.g., due to management station or network failure) the conceptual row will be left in the `notInService' or `notReady' state, consuming resources indefinitely. The agent must detect conceptual rows that have been in either state for an abnormally long period of time and remove them. It is the responsibility of the DESCRIPTION clause of the status column to indicate what an abnormally long period of time would be. This period of time should be long enough to allow for human response time (including `think time') between the creation of the conceptual row and the setting of the status to `active'. In the absence of such information in the DESCRIPTION clause, it is suggested that this period be approximately 5 minutes in length. This removal action applies not only to newly-created rows, but also to previously active rows which are set to, and left in, the notInService state for a prolonged period exceeding that which is considered normal for such a conceptual row.

Conceptual Row Suspension
When a conceptual row is `active', the management station may issue a management protocol set operation which sets the instance of the status column to `notInService'. If the agent is unwilling to do so, the set operation fails with an error of `wrongValue'. Otherwise, the conceptual row is taken out of service, and a `noError' response is returned. It is the responsibility of the DESCRIPTION clause of the status column to indicate under what circumstances the status column should be taken out of service (e.g., in order for the value of some other column of the same conceptual row to be modified).

Conceptual Row Deletion
For deletion of conceptual rows, a management protocol set operation is issued which sets the instance of the status column to `destroy'. This request may be made regardless of the current value of the status column (e.g., it is possible to delete conceptual rows which are either `notReady', `notInService' or `active'). If the operation succeeds, then all instances associated with the conceptual row are immediately removed.

SYNTAX INTEGER
{
   active(1),
   notInService(2),
   notReady(3),
   createAndGo(4),
   createAndWait(5),
   destroy(6)
TimeStamp ::= TEXTUAL-CONVENTION  
STATUS current 
DESCRIPTION  
"The value of the sysUpTime object at which a specific occurrence happened. The specific occurrence must be defined in the description of any object defined using this type."
SYNTAX TimeTicks

TimeInterval ::= TEXTUAL-CONVENTION  
STATUS current 
DESCRIPTION  
"A period of time, measured in units of 0.01 seconds."
SYNTAX INTEGER (0..2147483647)

DateAndTime ::= TEXTUAL-CONVENTION  
DISPLAY-HINT  
"2d-1d-1d,1d:1d:1d.1d,1a1d:1d"
STATUS current 
DESCRIPTION  
"A date-time specification.  
field  octets  contents                  range  
-----  ------  --------                  -----  
1      1-2   year                      0..65536  
2       3    month                     1..12  
3       4    day                       1..31  
4       5    hour                      0..23  
5       6    minutes                   0..59  
6       7    seconds                   0..60  
    (use 60 for leap-second)  
7       8    deci-seconds              0..9  
8       9    direction from UTC        '+' / '-'  
9      10    hours from UTC            0..11  
10      11    minutes from UTC          0..59
For example, Tuesday May 26, 1992 at 1:30:15 PM EDT would be displayed as:  
1992-5-26,13:30:15.0,-4:0
Note that if only local time is known, then timezone information (fields 8-10) is not present."
SYNTAX OCTET STRING (SIZE (8 | 11))

StorageType ::= TEXTUAL-CONVENTION  
STATUS current 
DESCRIPTION  
"Describes the memory realization of a conceptual row. A row which is volatile(2) is lost upon reboot. A row which is either nonVolatile(3), permanent(4) or readOnly(5), is backed up by stable storage. A row which is permanent(4) can be changed but not deleted. A row which is readOnly(5) cannot be changed nor deleted.  
If the value of an object with this syntax is either permanent(4) or readOnly(5), it cannot be modified. Conversely, if the value is either other(1), volatile(2) or nonVolatile(3), it cannot be modified to be permanent(4) or readOnly(5).  
Every usage of this textual convention is required to specify the columnar objects which a permanent(4) row must at a minimum allow to be writable."
SYNTAX INTEGER  
{ other(1), volatile(2), nonVolatile(3), permanent(4), readOnly(5) }

TDomain ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
"Denotes a kind of transport service.

Some possible values, such as snmpUDPDoman, are defined in 'Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)'."

SYNTAX OBJECT IDENTIFIER

TAddress ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION
"Denotes a transport service address.

For snmpUDPDoman, a TAddress is 6 octets long, the initial 4 octets containing the IP-address in network-byte order and the last 2 containing the UDP port in network-byte order. Consult 'Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)' for further information on snmpUDPDoman."

SYNTAX OCTET STRING (SIZE (1..255))

END