

Network Working Group
Request for Comments: 2513
Category: Standards Track

K. McCloghrie
Cisco Systems, Inc.
J. Heenanen
Telia Finland, Inc.
W. Greene
MCI Telecommunications Corp.
A. Prasad
Cisco Systems, Inc.
February 1999

Managed Objects for Controlling the Collection
and Storage of Accounting Information for
Connection-Oriented Networks

Status of this Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

Table of Contents

1 Introduction	2
2 The SNMP Network Management Framework	2
3 Overview	3
3.1 Operational Model	3
3.2 Selection of Accounting Data	5
3.3 Format of Collection File	6
4 Definitions	9
5 Acknowledgements	25
6 References	25
7 Security Considerations	27
8 IANA Considerations	27
9 Authors' Addresses	28
10 Full Copyright Statement	29

1. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in the Internet community. In particular, it describes managed objects used for controlling the collection and storage of accounting information for connection-oriented networks such as ATM. The accounting data is collected into files for later retrieval via a file transfer protocol. For information on data which can be collected for ATM networks, see [19].

2. The SNMP Network Management Framework

The SNMP Management Framework presently consists of five major components:

- o An overall architecture, described in RFC 2271 [1].
- o Mechanisms for describing and naming objects and events for the purpose of management. The first version of this Structure of Management Information (SMI) is called SMIV1 and described in STD 16, RFC 1155 [2], STD 16, RFC 1212 [3] and RFC 1215 [4]. The second version, called SMIV2, is described in RFC 1902 [5], RFC 1903 [6] and RFC 1904 [7].
- o Message protocols for transferring management information. The first version of the SNMP message protocol is called SNMPv1 and described in STD 15, RFC 1157 [8]. A second version of the SNMP message protocol, which is not an Internet standards track protocol, is called SNMPv2c and described in RFC 1901 [9] and RFC 1906 [10]. The third version of the message protocol is called SNMPv3 and described in RFC 1906 [10], RFC 2272 [11] and RFC 2274 [12].
- o Protocol operations for accessing management information. The first set of protocol operations and associated PDU formats is described in STD 15, RFC 1157 [8]. A second set of protocol operations and associated PDU formats is described in RFC 1905 [13].
- o A set of fundamental applications described in RFC 2273 [14] and the view-based access control mechanism described in RFC 2275 [15].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are defined using the mechanisms defined in the SMI.

This memo specifies a MIB module that is compliant to the SMIV2. A MIB conforming to the SMIV1 can be produced through the appropriate translations. The resulting translated MIB must be semantically equivalent, except where objects or events are omitted because no translation is possible (e.g., use of Counter64). Some machine readable information in SMIV2 will be converted into textual descriptions in SMIV1 during the translation process. However, this loss of machine readable information is not considered to change the semantics of the MIB.

3. Overview

In some connection-oriented network environments, there is a need for the network administrator to be able to collect accounting data on the usage of bandwidth/resources by connections (e.g., ATM connections) within the network. Data collection should be available for switched virtual connections (SVCs and SVPs), and permanent virtual connections (PVCs and PVPs), including soft-permanent virtual connections (SPVCCs and SPVPCs). This need exists for ATM networks, and may well exist for other connection-oriented networks, such as Frame Relay.

The potential quantity of such accounting information is such that it is not, in general, feasible to retrieve the information via SNMP. A better method is to store the collected accounting information in a file which can be subsequently retrieved via a file transfer protocol.

It is, however, appropriate to provide management control of the selection and collection of such accounting data via SNMP. This memo describes a MIB module which provides such control in a manner independent of the type of network. One or more other documents provide definitions of particular items of accounting data which can be selected; for example, a particular set of data items which can be collected for ATM networks is specified in [19].

3.1. Operational Model

The requirement is for switches (e.g., ATM switches) to collect data concerning the connections which are routed across some subset of their interfaces (e.g., ATM UNI and/or NNI interfaces). The collected data is stored into one or more "files". The use of multiple files allows, for example, the data collected for PVCs to be different from that collected for SVCs.

In order to retrieve the data currently being stored in a file, the administrator instructs the switch to terminate the collection of data into that file, and start collecting data into a new file.

After this operation, the data in the old file is available for retrieval via file transfer.

A collection file is defined to have a maximum size. When the size of the file currently being collected exceeds a threshold percentage of that maximum size, an SNMP notification (e.g., a trap) can be optionally generated. An SNMP notification might also be generated if the file reaches its maximum size.

The accounting data collected for each connection consists of a set of objects and their values. The set of objects and their values are collected on one or more of the following occasions:

- (1) on the release (termination) of a connection optionally including failed connection attempts;
- (2) for each active connection (having a particular minimum age) on a periodic basis;
- (3) for each active connection (having a particular minimum age) when so commanded by a management application.

While collecting data to be stored in a particular file, the same set of objects is collected for each connection on each occasion. Having the same set of objects stored on each occasion allows the optimization of storing only the values of those objects. This results in a significantly smaller file size, since it allows the names of the objects to be stored once and only once at the beginning of the file, rather than having to store every value as a (name, value) pair.

Two modes of agent behaviour are allowed on the event of a file reaching its maximum size:

- (1) management application in control:

The agent does not automatically swap to a new file; rather, it discards newly collected data until the management application subsequently instructs it to swap to a new file. Before swapping to a new file, the name of the file into which data is currently being collected is an implementation issue of no concern to an NM application; after swapping to a new file, the name of the file available for retrieval is as specified by the controlling MIB objects. This behaviour allows the application to know exactly how many files need to be retrieved and their names without having to perform any type of file directory operation, but also results in the possibility that data will be discarded if the application does not instruct the agent to swap

within the required time frame.

(2) agent automatically swaps to new file:

The agent terminates collection into the current (full) file, and begins collecting data into a new version of the same base file name. This behaviour aims to avoid loss of data by assuming that additional storage space is actually available to create a new version of the file. To support this behaviour, files are named using suffixes, such that when the current version of the file becomes full, the agent begins collecting data into a file with the same base file-name but with an incremented (or otherwise modified) suffix. This requires the application to perform file directory operations prior to retrieving completed files in order to know how many and which suffixes have been used.

With either behaviour, any completed file must be an integral number of connection records (see below). When a file reaches its maximum size, collection into that file is terminated either immediately before or immediately after storing the whole of the current connection record into the file. The former causes the file to be just less than its maximum size, and the latter causes the file to be just greater than its maximum size.

3.2. Selection of Accounting Data

The items of accounting data to be collected are specified as a set of objects. Which objects are contained in such a set is selectable by an administrator through the specification of one or more (subtree, list) tuples, where the set of objects to be collected is the union of the subsets specified by each tuple:

'subtree' specifies an OBJECT IDENTIFIER value such that every object in the subset is named by the subtree's value appended with a single additional sub-identifier.

'list' specifies an OCTET STRING value, such that if the N-th bit of the string's value is set then the subset contains the object named by appending N as a single additional sub-identifier to the subtree.

The rationale for defining each subset as a (subtree,list) tuple is that one and only one OBJECT IDENTIFIER and one OCTET STRING is needed to define the subset of objects. This simplifies the MIB mechanisms needed for selection: an NM application needs to create only one conceptual row in a MIB table for each subset (rather than needing to create a conceptual row in a table for each and every

object in the set).

The number of tuples supported by a particular switch is an implementation choice. One possibility is to support two (subtree, list) tuples so that one such tuple can specify a standard 'subtree' (e.g., the atmAcctngDataObjects subtree defined in [19]), and the second tuple can specify an enterprise-specific 'subtree'; this would allow the selected set of objects to be the union of a set of standard objects and a set of enterprise-defined objects.

3.3. Format of Collection File

A collection file generated by this process contains the values of MIB objects defined using the SMIV2. The standard way to encode the values of SNMP MIB objects in a device-independent manner is through the use of ASN.1's Basic Encoding Rules (BER) [18]. Thus, the standard format of an accounting file is defined here using the same adapted subset of ASN.1 [17] as the SMIV2.

The file consists of a set of header information followed by a sequence of zero or more collection records. The header information identifies (via sysName [16]) the switch which collected the data, the date and time at which the collection in to this file started, and the sequence of one or more (subtree, list) tuples identifying the objects whose values are contained in each connection record. The header information also includes a textual description of the data contained in the file.

Each connection record contains a sequence of values for each identified tuple, in the same order as the tuples are identified in the header information. For each tuple, the sequence of values are in ascending order of the sub-identifier which identifies them within the subtree.

Formally, an accounting file is an ASN.1 value with the following syntax:

```
File ::=
  [1]
    IMPLICIT SEQUENCE {
      -- header information
      sysName          -- name of the switch
        DisplayString,
      description      -- textual description of the collection
        DisplayString,
      startTime        -- start time of the collection
```

```

    DateAndTime,

    SEQUENCE OF {          -- sequence of (subtree, list) tuples
        SEQUENCE {
            subtree
            OBJECT IDENTIFIER,
            list
            OCTET STRING
        }
    }

    -- sequence of connection records
    SEQUENCE OF {
        -- each record containing a sequence
        SEQUENCE OF {      -- per identified tuple
            SEQUENCE OF {  -- each per-tuple sequence containing
                value       -- a sequence of object values
                ObjectSyntax
            }
        }
    }
}

```

where:

- (1) the value of the sysName component is that of the sysName object in the System group [16].
- (2) each (subtree, list) specifies the set of objects contained in that tuple's sequence within each and every connection record.
- (3) the tuples' sequences within each connection record occur in the same order as the (subtree, list) tuples occur in the header information.
- (4) the object values within each connection record occur in the same order as they are represented by the bits in the corresponding list value.
- (5) ObjectSyntax is defined by the SMIV2 [5].
- (6) One particular category of object values deserves special attention: an object defined to hold the checksum value of an accounting record (e.g., atmAcctngRecordCrc16, defined in [19]). An object in this category will generally have a SYNTAX of a fixed-length OCTET STRING, and have its value initialized to the string of all zeros when composing the accounting record containing it, with the location of these zeros being saved.

Once the record is generated, the checksum is calculated over the whole connection record (including the starting SEQUENCE OF and the trailing end-of-contents octets, if used), and then the zeros are overwritten (at the saved location) by the calculated value of the checksum.

The encoding of the above syntax using the Basic Encoding Rules is the same as defined by the SNMPv2 [10], with the following exception:

- when encoding the length field for a structured type, i.e., a SEQUENCE or SEQUENCE OF, the indefinite form encoding is permitted.

For example, the file containing the data:

```
[1] IMPLICIT SEQUENCE      a1 80
    OCTET STRING           04 09 73 77 69 74 63 68 2d 31 32
    OCTET STRING           04 0a 41 63 63 6f 75 6e 74 69 6e 67
    OCTET STRING           04 08 07 cc 07 14 10 05 00 00
    SEQUENCE OF            30 0e
        SEQUENCE           30 0c
            OBJECT IDENTIFIER 06 07 2b 06 01 03 7f 01 01
            OCTET STRING     04 01 c0
        SEQUENCE OF         30 80
            SEQUENCE OF      30 08
                SEQUENCE OF  30 06
                    INTEGER   02 01 00
                    INTEGER   02 01 21
                SEQUENCE OF   30 08
                    SEQUENCE OF 30 06
                        INTEGER  02 01 00
                        INTEGER  02 01 22
            end-of-contents  00 00
    end-of-contents        00 00
```

contains two connection records, each containing one tuple listing two (integer) data items in a (fictitious) subtree: 1.3.6.1.3.127.1.1. Its header indicates it's for "switch-12", with description "Accounting", and was collected at 16:05:00 on 20 July 1996.

As well as the standard format defined above, the MIB allows other enterprise-specific formats to be used.

4. Definitions

ACCOUNTING-CONTROL-MIB DEFINITIONS ::= BEGIN

IMPORTS

```

MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE,
mib-2, Integer32                                FROM SNMPv2-SMI
TEXTUAL-CONVENTION, RowStatus, TestAndIncr,
DisplayString, TruthValue                        FROM SNMPv2-TC
MODULE-COMPLIANCE, OBJECT-GROUP, NOTIFICATION-GROUP
                                                FROM SNMPv2-CONF
ifIndex                                          FROM IF-MIB;

```

accountingControlMIB MODULE-IDENTITY

```

LAST-UPDATED "9809281000Z"
ORGANIZATION "IETF ATOM MIB Working Group"
CONTACT-INFO "Keith McCloghrie
               Cisco Systems, Inc.
               170 West Tasman Drive,
               San Jose CA 95134-1706.
               Phone: +1 408 526 5260
               Email: kzm@cisco.com"

```

DESCRIPTION

"The MIB module for managing the collection and storage of accounting information for connections in a connection-oriented network such as ATM."

::= { mib-2 60 }

acctngMIBObjects OBJECT IDENTIFIER ::= { accountingControlMIB 1 }

acctngSelectionControl OBJECT IDENTIFIER ::= { acctngMIBObjects 1 }

acctngFileControl OBJECT IDENTIFIER ::= { acctngMIBObjects 2 }

acctngInterfaceControl OBJECT IDENTIFIER ::= { acctngMIBObjects 3 }

acctngTrapControl OBJECT IDENTIFIER ::= { acctngMIBObjects 4 }

-- Textual Conventions

DataCollectionSubtree ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The subtree component of a (subtree, list) tuple. Such a (subtree, list) tuple defines a set of objects and their values to be collected as accounting data for a connection. The subtree specifies a single OBJECT IDENTIFIER value such that each object in the set is named by the subtree value

appended with a single additional sub-identifier."
SYNTAX OBJECT IDENTIFIER

DataCollectionList ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"The list component of a (subtree, list) tuple. Such a (subtree, list) tuple defines a set of objects and their values to be collected as accounting data for a connection. The subtree specifies a single OBJECT IDENTIFIER value such that each object in the set is named by the subtree value appended with a single additional sub-identifier. The list specifies a set of data items, where the presence of an item in the list indicates that the item is (to be) present in the data collected for a connection; the absence of an item from the list indicates that the item is not (to be) present in the data collected for a connection. Each data item is represented by an integer which when appended (as an additional sub-identifier) to the OBJECT IDENTIFIER value of the subtree identified by the tuple, is the name of an object defining that data item (its description and its syntax).

The list is specified as an OCTET STRING in which each data item is represented by a single bit, where data items 1 through 8 are represented by the bits in the first octet, data items 9 through 16 by the bits in the second octet, etc. In each octet, the lowest numbered data item is represented by the most significant bit, and the highest numbered data item by the least significant bit. A data item is present in the list when its bit is set, and absent when its bit is reset. If the length of an OCTET STRING value is too short to represent one or more data items defined in a subtree, then those data items are absent from the set identified by the tuple of that subtree and that OCTET STRING value."

SYNTAX OCTET STRING (SIZE(0..8))

FileIndex ::= TEXTUAL-CONVENTION

STATUS current

DESCRIPTION

"An arbitrary integer value identifying a file into which accounting data is being collected."

SYNTAX Integer32 (1..65535)

-- The Accounting Information Selection table

acctngSelectionTable OBJECT-TYPE

SYNTAX SEQUENCE OF AcctngSelectionEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION

"A list of accounting information selection entries.

Note that additions, modifications and deletions of entries in this table can occur at any time, but such changes only take effect on the next occasion when collection begins into a new file. Thus, between modification and the next 'swap', the content of this table does not reflect the current selection."

::= { acctngSelectionControl 1 }

acctngSelectionEntry OBJECT-TYPE

SYNTAX AcctngSelectionEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION

"An entry identifying an (subtree, list) tuple used to select a set of accounting information which is to be collected."

INDEX { acctngSelectionIndex }

::= { acctngSelectionTable 1 }

AcctngSelectionEntry ::=

SEQUENCE {
 acctngSelectionIndex Integer32,
 acctngSelectionSubtree DataCollectionSubtree,
 acctngSelectionList DataCollectionList,
 acctngSelectionFile FileIndex,
 acctngSelectionType BITS,
 acctngSelectionRowStatus RowStatus
 }

acctngSelectionIndex OBJECT-TYPE

SYNTAX Integer32 (1..65535)
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION

"An arbitrary integer value which uniquely identifies a tuple stored in this table. This value is required to be the permanent 'handle' for an entry in this table for as long as that entry exists, including across restarts and power outages."

::= { acctngSelectionEntry 1 }

acctngSelectionSubtree OBJECT-TYPE

SYNTAX DataCollectionSubtree

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The combination of acctngSelectionSubtree and acctngSelectionList specifies one (subtree, list) tuple which is to be collected."

::= { acctngSelectionEntry 2 }

acctngSelectionList OBJECT-TYPE

SYNTAX DataCollectionList

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The combination of acctngSelectionSubtree and acctngSelectionList specifies one (subtree, list) tuple which is to be collected."

::= { acctngSelectionEntry 3 }

acctngSelectionFile OBJECT-TYPE

SYNTAX FileIndex

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"An indication of the file into which the accounting information identified by this entry is to be stored. If there is no conceptual row in the acctngFileTable for which the value of acctngFileIndex has the same value as this object, then the information selected by this entry is not collected."

::= { acctngSelectionEntry 4 }

acctngSelectionType OBJECT-TYPE

SYNTAX BITS { svcIncoming(0),
 svcOutgoing(1),
 svpIncoming(2),
 svpOutgoing(3),
 pvc(4),
 pvp(5),
 spvcOriginator(6),
 spvcTarget(7),
 spvpOriginator(8),
 spvpTarget(9) }

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"Indicates the types of connections for which the

```

        information selected by this entry are to be collected."
DEFVAL      { { svcIncoming, svcOutgoing,
                svpIncoming, svpOutgoing } }
 ::= { acctngSelectionEntry 5 }

```

acctngSelectionRowStatus OBJECT-TYPE

```

SYNTAX      RowStatus
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION

```

"The status of this conceptual row. An agent may refuse to create new conceptual rows and/or modify existing conceptual rows, if such creation/modification would cause multiple rows to have the same values of acctngSelectionSubtree and acctngSelectionList.

A conceptual row can not have the status of 'active' until values have been assigned to the acctngSelectionSubtree, acctngSelectionList and acctngSelectionFile columnar objects within that row.

An agent must not refuse to change the values of the acctngSelectionSubtree, acctngSelectionList and acctngSelectionFile columnar objects within a conceptual row even while that row's status is 'active'. Similarly, an agent must not refuse to destroy an existing conceptual row while the file referenced by that row's instance of acctngSelectionFile is in active use, i.e., while the corresponding instance of acctngFileRowStatus has the value 'active'. However, such changes only take effect upon the next occasion when collection begins into a new (version of the) file."

```
 ::= { acctngSelectionEntry 6 }
```

-- The Accounting File table

acctngFileTable OBJECT-TYPE

```

SYNTAX      SEQUENCE OF AcctngFileEntry
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION

```

"A list of files into which accounting information is to be stored."

```
 ::= { acctngFileControl 1 }
```

acctngFileEntry OBJECT-TYPE

```

SYNTAX      AcctngFileEntry
MAX-ACCESS  not-accessible

```

```

STATUS      current
DESCRIPTION
    "An entry identifying a file into which accounting
    information is to be collected."
INDEX       { acctngFileIndex }
 ::= { acctngFileTable 1 }

```

```

AcctngFileEntry ::=
SEQUENCE {
    acctngFileIndex          FileIndex,
    acctngFileName           DisplayString,
    acctngFileNameSuffix     DisplayString,
    acctngFileDescription    DisplayString,
    acctngFileCommand        INTEGER,
    acctngFileMaximumSize    Integer32,
    acctngFileCurrentSize    Integer32,
    acctngFileFormat         INTEGER,
    acctngFileCollectMode    BITS,
    acctngFileCollectFailedAttempts BITS,
    acctngFileInterval       Integer32,
    acctngFileMinAge         Integer32,
    acctngFileRowStatus      RowStatus
}

```

```

acctngFileIndex OBJECT-TYPE
SYNTAX      FileIndex
MAX-ACCESS  not-accessible
STATUS      current
DESCRIPTION
    "A unique value identifying a file into which accounting
    data is to be stored.  This value is required to be the
    permanent 'handle' for an entry in this table for as long as
    that entry exists, including across restarts and power
    outages."
 ::= { acctngFileEntry 1 }

```

```

acctngFileName OBJECT-TYPE
SYNTAX      DisplayString (SIZE(1..32))
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
    "The name of the file into which accounting data is to be
    stored.  If files are named using suffixes, then the name of
    the current file is the concatenation of acctngFileName and
    acctngFileNameSuffix.

    An agent will respond with an error (e.g., 'wrongValue') to
    a management set operation which attempts to modify the

```

value of this object to the same value as already held by another instance of acctngFileName. An agent will also respond with an error (e.g., 'wrongValue') if the new value is invalid for use as a file name on the local file system (e.g., many file systems do not support white space embedded in file names).

The value of this object can not be modified while the corresponding instance of acctngFileRowStatus is 'active'."

```
::= { acctngFileEntry 2 }
```

acctngFileNameSuffix OBJECT-TYPE

SYNTAX DisplayString (SIZE(0..8))

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The suffix, if any, of the name of a file into which accounting data is currently being stored. If suffixes are not used, then the value of this object is the zero-length string. Note that if a separator, such as a period, is used in appending the suffix to the file name, then that separator appears as the first character of this value."

```
::= { acctngFileEntry 3 }
```

acctngFileDescription OBJECT-TYPE

SYNTAX DisplayString

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The textual description of the accounting data which will be stored (on the next occasion) when header information is stored in the file. The value of this object may be modified at any time."

DEFVAL { "" }

```
::= { acctngFileEntry 4 }
```

acctngFileCommand OBJECT-TYPE

SYNTAX INTEGER {

-- the following two values are states:

-- they may be read but not written

idle(1),

cmdInProgress(2),

-- the following two values are actions:

-- they may be written, but are never read

swapToNewFile(3),

collectNow(4)

}

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"A control object for the collection of accounting data. When read the value is either 'idle' or 'cmdInProgress'. Writing a value is only allowed when the current value is 'idle'. When a value is successfully written, the value changes to 'cmdInProgress' until completion of the action, at which time the value reverts to 'idle'. Actions are invoked by writing the following values:

'swapToNewFile' - the collection of data into the current file is terminated, and collection continues into a new (version of the) file.

'collectNow' - the agent creates and stores a connection record into the current file for each active connection having a type matching acctngSelectionType and an age greater than acctngFileMinAge."

DEFVAL { idle }
::= { acctngFileEntry 5 }

acctngFileMaximumSize OBJECT-TYPE

SYNTAX Integer32 (100..2147483647)

UNITS "bytes"

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The maximum size of the file (including header information). When the file of collected data reaches this size, either the agent automatically swaps to a new version (i.e., a new value acctngFileNameSuffix) of the file, or new records are discarded. Since a file must contain an integral number of connection records, the actual maximum size of the file may be just less OR Just greater than the value of this object.

The value of this object can not be modified while the corresponding instance of acctngFileRowStatus is 'active'. The largest value of the maximum file size in some agents will be less than 2147483647 bytes."

DEFVAL { 5000000 }
::= { acctngFileEntry 6 }

acctngFileCurrentSize OBJECT-TYPE

SYNTAX Integer32 (0..2147483647)

UNITS "bytes"

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"The current size of the file into which data is currently being collected, including header information."

::= { acctngFileEntry 7 }

acctngFileFormat OBJECT-TYPE

SYNTAX INTEGER { other(1), ber(2) }

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"An indication of the format in which the accounting data is to be stored in the file. If the value is modified, the new value takes effect after the next 'swap' to a new file. The value ber(2) indicates the standard format."

DEFVAL { ber }

::= { acctngFileEntry 8 }

acctngFileCollectMode OBJECT-TYPE

SYNTAX BITS { onRelease(0), periodically(1) }

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"An indication of when accounting data is to be written into this file. Note that in addition to the occasions indicated by the value of this object, an agent always writes information on appropriate connections to the file when the corresponding instance of acctngFileCommand is set to 'collectNow'.

- 'onRelease' - whenever a connection (or possibly, connection attempt) is terminated, either through a Release message or through management removal, information on that connection is written.

- 'periodically' - information on appropriate connections is written on the expiry of a periodic timer,

This value may be modified at any time."

DEFVAL { { onRelease } }

::= { acctngFileEntry 9 }

acctngFileCollectFailedAttempts OBJECT-TYPE

SYNTAX BITS { soft(0), regular(1) }

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"An indication of whether connection data is to be collected

for failed connection attempts when the value of the corresponding instance of `acctngFileCollectMode` includes 'onRelease'. The individual values have the following meaning:

'soft' - indicates that connection data is to be collected for failed Soft PVCs/PVPs which originate or terminate at the relevant interface.

'regular' - indicates that connection data is to be collected for failed SVCs, including Soft PVCs/PVPs not originating or terminating at the relevant interface.

This value may be modified at any time."

```
DEFVAL      { { soft, regular } }
::= { acctngFileEntry 10 }
```

`acctngFileInterval` OBJECT-TYPE

```
SYNTAX      Integer32 (60..86400)
UNITS       "seconds"
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
```

"The number of seconds between the periodic collections of accounting data when the value of the corresponding instance of `acctngFileCollectMode` includes 'periodically'. Some agents may impose restrictions on the range of this interval. This value may be modified at any time."

```
DEFVAL      { 3600 }
::= { acctngFileEntry 11 }
```

`acctngFileMinAge` OBJECT-TYPE

```
SYNTAX      Integer32 (60..86400)
UNITS       "seconds"
MAX-ACCESS  read-create
STATUS      current
DESCRIPTION
```

"The minimum age of a connection, as used to determine the set of connections for which data is to be collected at the periodic intervals and/or when `acctngFileCommand` is set to 'collectNow'. The age of a connection is the elapsed time since it was last installed.

When the periodic interval expires for a file or when `acctngFileCommand` is set to 'collectNow', accounting data is collected and stored in the file for each connection having a type matching `acctngSelectionType` and whose age at that time is greater than the value of `acctngFileMinAge`

associated with the file. This value may be modified at any time."

DEFVAL { 3600 }
 ::= { acctngFileEntry 12 }

acctngFileRowStatus OBJECT-TYPE

SYNTAX RowStatus

MAX-ACCESS read-create

STATUS current

DESCRIPTION

"The status of this conceptual row.

This object can not be set to 'active' until a value has been assigned to the corresponding instance of acctngFileName. Collection of data into the file does not begin until this object has the value 'active' and one or more (active) instances of acctngSelectionFile refer to it. If this value is modified after a collection has begun, collection into this file terminates and a new (or new version of the) file is immediately made ready for future collection (as if acctngFileCommand had been set to 'swapToNewFile'), but collection into the new (or new version of the) file does not begin until the value is subsequently set back to active."

::= { acctngFileEntry 13 }

-- Overall Control

acctngAdminStatus OBJECT-TYPE

SYNTAX INTEGER { enabled(1), disabled(2) }

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"A control object to indicate the administratively desired state of the collection of accounting records across all interfaces.

Modifying the value of acctngAdminStatus to 'disabled' does not remove or change the current configuration as represented by the active rows in the acctngSelectionTable, acctngFileTable and acctngInterfaceTable tables."

::= { acctngInterfaceControl 1 }

acctngOperStatus OBJECT-TYPE

SYNTAX INTEGER { enabled(1), disabled(2) }

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"A status object to indicate the operational state of the collection of accounting records across all interfaces.

When the value of `acctngAdminStatus` is modified to be 'enabled', the value of this object will change to 'enabled' providing it is possible to begin collecting accounting records.

When the value of `acctngAdminStatus` is modified to be 'disabled', the value of this object will change to 'disabled' as soon as the collection of accounting records has terminated."

::= { `acctngInterfaceControl` 2 }

`acctngProtection` OBJECT-TYPE

SYNTAX TestAndIncr

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"A control object to protect against duplication of control commands. Over some transport/network protocols, it is possible for SNMP messages to get duplicated. Such duplication, if it occurred at just the wrong time could cause serious disruption to the collection and retrieval of accounting data, e.g., if a SNMP message setting `acctngFileCommand` to 'swapToNewFile' were to be duplicated, a whole file of accounting data could be lost.

To protect against such duplication, a management application should retrieve the value of this object, and include in the Set operation needing protection, a variable binding which sets this object to the retrieved value."

::= { `acctngInterfaceControl` 3 }

`acctngAgentMode` OBJECT-TYPE

SYNTAX INTEGER { `swapOnCommand`(1), `swapOnFull`(2) }

MAX-ACCESS read-only

STATUS current

DESCRIPTION

"An indication of the behaviour mode of the agent when a file becomes full:

'`swapOnCommand`' - the agent does not automatically swap to a new file; rather, it discards newly collected data until a management application subsequently instructs it to swap to a new file.

'`swapOnFull`' - the agent terminates collection into the

```

                                current file as and when that file becomes full."
 ::= { acctngInterfaceControl 4 }

```

```
-- Per-interface control table
```

```
acctngInterfaceTable OBJECT-TYPE
```

```
    SYNTAX      SEQUENCE OF AcctngInterfaceEntry
```

```
    MAX-ACCESS  not-accessible
```

```
    STATUS      current
```

```
    DESCRIPTION
```

```
        "A table controlling the collection of accounting data on
        specific interfaces of the switch."
```

```
 ::= { acctngInterfaceControl 5 }
```

```
acctngInterfaceEntry OBJECT-TYPE
```

```
    SYNTAX      AcctngInterfaceEntry
```

```
    MAX-ACCESS  not-accessible
```

```
    STATUS      current
```

```
    DESCRIPTION
```

```
        "An entry which controls whether accounting data is to be
        collected on an interface. The types of interfaces which
        are represented in this table is implementation-specific."
```

```
    INDEX      { ifIndex }
```

```
 ::= { acctngInterfaceTable 1 }
```

```
AcctngInterfaceEntry ::=
```

```
    SEQUENCE {
```

```
        acctngInterfaceEnable      TruthValue
```

```
    }
```

```
acctngInterfaceEnable OBJECT-TYPE
```

```
    SYNTAX      TruthValue
```

```
    MAX-ACCESS  read-write
```

```
    STATUS      current
```

```
    DESCRIPTION
```

```
        "Indicates whether the collection of accounting data is
        enabled on this interface."
```

```
 ::= { acctngInterfaceEntry 1 }
```

```
-- Objects for controlling the use of Notifications
```

```
acctngControlTrapThreshold OBJECT-TYPE
```

```
    SYNTAX      INTEGER (0..99)
```

```
    MAX-ACCESS  read-write
```

```
    STATUS      current
```

```
    DESCRIPTION
```

```
        "A percentage of the maximum file size at which a 'nearly-
```

full' trap is generated. The value of 0 indicates that no
 'nearly-full' trap is to be generated."
 ::= { acctngTrapControl 1 }

acctngControlTrapEnable OBJECT-TYPE

SYNTAX TruthValue

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"An indication of whether the acctngFileNearlyFull and
 acctngFileFull traps are enabled."

::= { acctngTrapControl 2 }

-- notifications

acctngNotifications OBJECT IDENTIFIER ::= { accountingControlMIB 2 }

acctngNotifyPrefix OBJECT IDENTIFIER ::= { acctngNotifications 0 }

acctngFileNearlyFull NOTIFICATION-TYPE

OBJECTS { acctngFileName,
 acctngFileMaximumSize,
 acctngControlTrapThreshold,
 acctngFileNameSuffix }

STATUS current

DESCRIPTION

"An indication that the size of the file into which
 accounting information is currently being collected has
 exceeded the threshold percentage of its maximum file size.
 This notification is generated only at the time of the
 transition from not-exceeding to exceeding."

::= { acctngNotifyPrefix 1 }

acctngFileFull NOTIFICATION-TYPE

OBJECTS { acctngFileName,
 acctngFileMaximumSize,
 acctngFileNameSuffix }

STATUS current

DESCRIPTION

"An indication that the size of the file into which
 accounting information is currently being collected has
 transistioned to its maximum file size. This notification
 is generated (for all values of acctngAgentMode) at the time
 of the transition from not-full to full. If acctngAgentMode
 has the value 'swapOnCommand', it is also generated
 periodically thereafter until such time as collection of

```

        data is no longer inhibited by the file full condition."
 ::= { acctngNotifyPrefix 2 }

```

```
-- conformance information
```

```

acctngConformance OBJECT IDENTIFIER ::= { accountingControlMIB 3 }
acctngGroups       OBJECT IDENTIFIER ::= { acctngConformance 1 }
acctngCompliances  OBJECT IDENTIFIER ::= { acctngConformance 2 }

```

```
acctngCompliance MODULE-COMPLIANCE
```

```
    STATUS    current
```

```
    DESCRIPTION
```

```
        "The compliance statement for switches which implement the
        Accounting Control MIB."
```

```
MODULE -- this module
```

```
    MANDATORY-GROUPS { acctngBasicGroup,
                        acctngNotificationsGroup }
```

```

OBJECT      acctngSelectionType
SYNTAX      BITS { svcIncoming(0), svcOutgoing(1) }
DESCRIPTION "The minimal requirement is collection for SVCs."

```

```

OBJECT      acctngSelectionRowStatus
MIN-ACCESS  read-only
DESCRIPTION "Write access is not required."

```

```

OBJECT      acctngFileName
MIN-ACCESS  read-only
DESCRIPTION "Write access is not required."

```

```

OBJECT      acctngFileCommand
MIN-ACCESS  read-only
DESCRIPTION "Write access is not required."

```

```

OBJECT      acctngFileFormat
SYNTAX      INTEGER { ber(2) }
MIN-ACCESS  read-only
DESCRIPTION "Only the standard format is required, and write
access is not required."

```

```

OBJECT      acctngFileMaximumSize
MIN-ACCESS  read-only
DESCRIPTION "Write access is not required."

```

```

OBJECT      acctngFileCollectMode
SYNTAX      BITS { onRelease(0) }

```

MIN-ACCESS read-only
 DESCRIPTION "The minimal requirement is for collection on connection release."

OBJECT acctngFileInterval
 MIN-ACCESS read-only
 DESCRIPTION "Write access is not required."

OBJECT acctngFileCollectFailedAttempts
 MIN-ACCESS read-only
 DESCRIPTION "Write access is not required."

OBJECT acctngFileRowStatus
 MIN-ACCESS read-only
 DESCRIPTION "Write access is not required."

::= { acctngCompliances 1 }

-- units of conformance

acctngBasicGroup OBJECT-GROUP
 OBJECTS { acctngSelectionSubtree, acctngSelectionList,
 acctngSelectionFile, acctngSelectionType,
 acctngSelectionRowStatus, acctngFileName,
 acctngFileNameSuffix, acctngFileDescription,
 acctngFileCommand, acctngFileMaximumSize,
 acctngFileCurrentSize, acctngFileRowStatus,
 acctngFileFormat, acctngFileCollectMode,
 acctngFileCollectFailedAttempts, acctngFileInterval,
 acctngFileMinAge,
 acctngAdminStatus, acctngOperStatus,
 acctngProtection, acctngAgentMode,
 acctngInterfaceEnable,
 acctngControlTrapThreshold,
 acctngControlTrapEnable
 }
 STATUS current
 DESCRIPTION
 "A collection of objects providing control of the basic
 collection of accounting data for connection-oriented
 networks."
 ::= { acctngGroups 1 }

acctngNotificationsGroup NOTIFICATION-GROUP
 NOTIFICATIONS { acctngFileNearlyFull, acctngFileFull }
 STATUS current
 DESCRIPTION


```
        "The notifications of events relating to controlling the  
        collection of accounting data."  
 ::= { acctngGroups 2 }
```

END

5. Acknowledgements

The comments of the IETF's ATOM MIB Working Group are acknowledged.

6. References

- [1] Harrington, D., Presuhn, R. and B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", RFC 2271, January 1998.
- [2] Rose, M. and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based Internets", STD 16, RFC 1155, May 1990.
- [3] Rose, M. and K. McCloghrie, "Concise MIB Definitions", STD 16, RFC 1212, March 1991.
- [4] Rose, M., "A Convention for Defining Traps for use with the SNMP", RFC 1215, March 1991.
- [5] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.
- [6] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Textual Conventions for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1903, January 1996.
- [7] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1904, January 1996.
- [8] Case, J., Fedor, M., Schoffstall, M. and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, May 1990.
- [9] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Introduction to Community-based SNMPv2", RFC 1901, January 1996.
- [10] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, January 1996.

- [11] Case, J., Harrington D., Presuhn R. and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", RFC 2272, January 1998.
- [12] Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", RFC 2274, January 1998.
- [13] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.
- [14] Levi, D., Meyer, P. and B. Stewart, "SNMPv3 Applications", RFC 2273, January 1998.
- [15] Wijnen, B., Presuhn, R. and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", RFC 2275, January 1998.
- [16] SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M. and S. Waldbusser, "Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1907, January 1996.
- [17] Information processing systems - Open Systems Interconnection, "Specification of Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8824, December 1987.
- [18] Information processing systems - Open Systems Interconnection, "Specification of Basic Encoding Rules for Abstract Syntax Notation One (ASN.1)", International Organization for Standardization, International Standard 8825, December 1987.
- [19] McCloghrie, K., Heinanen, J., Greene, W. and A. Prasad, "Accounting Information for ATM Networks", RFC 2512, February 1999.
- [20] Noto, M., Spiegel, E., and K. Tesink, "Definitions of Textual Conventions and OBJECT-IDENTITIES for ATM Management", RFC 2514, February 1999.

7. Security Considerations

The MIB defined in this memo controls and monitors the collection of accounting data. Care should be taken to prohibit unauthorized access to this control capability in order to prevent the disruption of data collection, possibly with fraudulent intent. Example of such disruption are disabling the collection of data, or causing the wrong set of data items to be collected.

SNMPv1 by itself is not a secure environment. Even if the network itself is secure (for example by using IPSec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB.

It is recommended that the implementers consider the security features as provided by the SNMPv3 framework. Specifically, the use of the User-based Security Model RFC 2274 [12] and the View-based Access Control Model RFC 2275 [15] is recommended.

It is then a customer/user responsibility to ensure that the SNMP entity giving access to an instance of this MIB, is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

8. IANA Considerations

Prior to publication of this memo as an RFC, IANA is requested to make a suitable OBJECT IDENTIFIER assignment.

9. Authors' Addresses

Keith McCloghrie
Cisco Systems, Inc.
170 West Tasman Drive,
San Jose CA 95134

Phone: +1 408 526 5260
EMail: kzm@cisco.com

Juha Heinanen
Telia Finland, Inc.
Myyrmaentie 2
01600 VANTAA
Finland

Phone +358 303 944 808
EMail: jh@telia.fi

Wedge Greene
MCI Telecommunications Corporation
901 International Parkway
Richardson, Texas 75081

Phone: 214-498-1232 or 972-729-1232
EMail: wedge.greene@mci.com

Anil Prasad
Cisco Systems, Inc.
170 West Tasman Drive,
San Jose CA 95134

Phone: 408 525-7209
EMail: aprasad@cisco.com

10. Full Copyright Statement

Copyright (C) The Internet Society (1999). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

