Bundle Protocol MIB
draft-sims-dtnrg-bpmib-02

Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols. In particular it defines objects for managing information about a Bundle Node - or simply a ‘Node’ within the scope of this document. More specifically, the managed objects for such a Node include: Node-specific information, registered Endpoint-Specific information, and generic CLA-Specific (Convergence Layer Adapter) information.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on October 22, 2012.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must
include Simplified BSD License text as described in Section 4.e of
the Trust Legal Provisions and are provided without warranty as
described in the Simplified BSD License.

Table of Contents

1. The Internet-Standard Management Framework .................. 3
2. Introduction .................................................... 3
3. Requirements Language ........................................ 3
4. Overview ........................................................ 3
4.1. Structure of the MIB Module ................................ 4
  4.1.1. Node-Specific Definitions ............................... 4
  4.1.2. Endpoint-Specific Definitions ........................... 5
  4.1.3. CLA-Specific Definitions ................................ 6
4.2. Design Decisions ............................................. 7
  4.2.1. Node and Endpoint Representation ....................... 7
  4.2.2. Extension Blocks .......................................... 8
  4.2.3. Convergence Layer Adapters .............................. 8
  4.2.4. Managing the Protocol Stack ............................. 8
  4.2.5. bpClType Number Standardization ........................ 9
  4.2.6. Reason for bpClaDirection ............................... 9
  4.2.7. Notifications ............................................ 10
  4.2.8. Statistical and Trended Data ........................... 10
4.3. Considerations for Future Designers ......................... 10
  4.3.1. Delay/Disruption Entity Management .................... 11
  4.3.2. CLA MIB Design ........................................... 11
  4.3.3. Notification MIB Design ................................. 12
  4.3.4. Extension Block MIB Design ............................. 12
5. Definitions ..................................................... 12
6. Acknowledgments ................................................ 49
7. IANA Considerations ............................................ 49
8. Security Considerations ........................................ 49
9. References ...................................................... 50
  9.1. Normative References ...................................... 50
  9.2. Informative References .................................... 50
1. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

2. Introduction

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols. In particular it defines objects for managing information about a Bundle Node [RFC5050] - or simply a 'Node' within the scope of this document. More specifically, the manageable objects for such a Node include: Node-Specific information, registered Endpoint-Specific information, and generic CLA-Specific (Convergence Layer Adapter) information.

3. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

4. Overview

The Bundle Protocol (BP) [RFC5050] was designed as part of the solution to issues encountered in Delay Tolerant Networking (DTN) environments [RFC4838], which are otherwise unsatisfied by the connection-oriented scheme used in today’s Internet. This memo contains the definitions for a Bundle Protocol MIB as well as rationale for certain implementation decisions. Also contained in this memo is a section dedicated to considerations for future designers of DTN-related MIBs. The MIB module defined in this memo contains object definitions for a single Node, for each Endpoint of which that Node is a member, and for the CLAs which that Node has access to use.
4.1. Structure of the MIB Module

Objects within the BP-MIB have been derived from RFC 5050 [RFC5050] or were designed to be used in the management of a BP Node, but are not explicitly derived from RFC 5050 [RFC5050]. These objects are broken down into three main groups:

- Node-Specific Definitions
- Endpoint-Specific Definitions
- CLA-Specific Definitions

4.1.1. Node-Specific Definitions

Within this document, objects that are referred to as ‘Node-Specific’ are those objects which relate only to the Node being managed. In other words, these objects do not relate to a Node’s relationship with any other entity (i.e. Endpoints and CLAs). Node-Specific objects include:

- A unique name given to the Node Instance (bpNodeID)
- The Bundle Protocol version being used by the managed Node (bpBundleVersion)
- The amount of storage available to the managed Node (bpAvailStorage)
- The last time a Node was restarted (bpLastUpTime)
- The number of locally generated Bundles for each priority type (bpBulkPriorGenBndls, bpNormPriorGenBndls, and bpExpPriorGenBndls)
- The number of locally generated Bytes for each priority type (bpBulkPriorGenBytes, bpNormPriorGenBytes, and bpExpPriorGenBytes)
- The number of locally queued Bundles for each priority type (bpBulkPriorQueuedBndls, bpNormPriorQueuedBndls, and bpExpPriorQueuedBndls)
- The number of locally queued Bytes for each priority type (bpBulkPriorQueuedBytes, bpNormPriorQueuedBytes, and bpExpPriorQueuedBytes)
- The number of locally queued Bundles for each type of retention constraint (bpReassemblyPendingBndls, bpDispatchPendingBndls, bpForwardPendingBndls, and bpCustodyAcceptedBndls)
The number of locally queued Bytes for each type of retention constraint (bpReassemblyPendingBytes, bpDispatchPendingBytes, bpForwardPendingBytes, and bpCustodyAcceptedBytes)

The number of reports received by this Node for each of the report types (bpBundleReceptions, bpBundleAcceptance, bpBundleForwarding, bpCustodySuccess, and bpBundleDelivery)

The number of times this Node took a specific action on (or failed to take action on) Bundles. Specifically, these actions involve: custody acceptance, forwarding, abandoning, discarding, deletion (for each or the reasons specified in RFC 5050 [RFC5050]), and fragmenting (bpCustodyFailureBndls, bpForwardFailureBndls, bpAbandonedDeliveryBndls, bpDiscardedBndls, bpFragmentedLocally, bpDelExpired, bpDelTransCancelled, bpDelDepStorage, bpDelDestUnintel, bpDelNoRoute, bpDelUntimelyContact, bpDelBlockUnintel, and bpDelNoAdditionalInfo)

The number of Bytes affected by this Node taking a specific action on (or failing to take action on) Bundles. Specifically, these actions involve: custody acceptance, forwarding, abandoning, discarding, and deletion (bpCustodyFailureBytes, bpForwardFailureBytes, bpAbandonedDeliveryBytes, and bpDiscardedBytes)

The number of fragments created by the managed Node (bpFragmentsCreated) and

Denotation of available BP Extension Blocks (bpExtBlockType and bpExtBlockDisplayName)

Each of these objects are detailed more precisely in the MIB definition’s description fields.

4.1.2. Endpoint-Specific Definitions

Within this document, objects that are referred to as ‘Endpoint-Specific’ are those objects which relate to a specific Endpoint in respect to the managed Node. In other words, the information managed by these objects is subject only to the Node’s view of the Endpoint (e.g. the current state of the Endpoint is "active" according to this Node OR the number of Bundles this Node has received through this Endpoint is 42). Endpoint-Specific objects are maintained within a conceptual table (with the exception of bpRegCount). The following objects fall under this category:

The number of currently registered Endpoints (bpRegCount)
The ID/address of the Endpoint Entry (bpEndpointID)

Denotation of the BP’s current state for this Endpoint (bpCurState)

Denotation of whether the Endpoint is a Singleton (bpIsSingleton)

Denotation of the Delivery Failure Action for this Endpoint (bpDeliveryFailureAction)

The number of times this Endpoint has experienced the following Bundle activities: received (for each priority type), delivered, sent, forwarded, and received duplicate of a queued bundle (bpBulkPriorRecvdBndls, bpNormPriorRecvdBndls, bpExpPriorRecvdBndls, bpBundlesDeliveredBndls, bpBundlesSentBndls, bpBundlesForwardedBndls, and bpDuplicatesBndls).

The number of Bytes affected due to this Endpoint experiencing the following Bundle activities: received (for each priority type), delivered, sent, forwarded, and received duplicate of a queued bundle (bpBulkPriorRecvdBytes, bpNormPriorRecvdBytes, bpExpPriorRecvdBytes, bpBundlesDeliveredBytes, bpBundlesSentBytes, bpBundlesForwardedBytes, and bpDuplicatesBytes) and

The number of Bundle Deletion Reports received on this Endpoint for each of the reasons specified in RFC 5050 [RFC5050] (bpDelExpiredReports, bpDelTransCancelledReports, bpDelDepStorageReports, bpDelDestUnintelReports, bpDelNoRouteReports, bpDelUntimelyContactReports, bpDelBlockUnintelReports, and bpDelNoAdditionalInfoReports).

Each of these objects are detailed more precisely in the MIB definition’s description fields.

4.1.3. CLA-Specific Definitions

Within this document, objects that are referred to as ‘CLA-Specific’ are those objects which relate to a specific CLA in respect to the managed Node. CLA-Specific objects are maintained within a conceptual table (with the exception of bpClaCount). The following objects fall under this category:

- The number of known CLAs (bpClaCount)
- Denotation of the CL type (e.g. UDP, TCP, LTP, etc) used by the CLA (bpClType)
o A human-readable name for the CLA assigned by the management agent (bpClDisplayName)

o A unique identifier for a specific CLA’s MIB instance (bpClID)

o The number of Bundles passed up/down the protocol stack from this CLA, to/from this BPA (bpClInBndls and bpClOutBndls)

o The number of Bytes passed up/down the protocol stack from this CLA, to/from this BPA (bpClInBytes and bpClOutBytes) and

o Denotation of the CLA’s transmission direction (bpClDirection)

Each of these objects are detailed more precisely in the MIB definition’s description fields.

4.2. Design Decisions

This section contains information regarding certain design choices made during the creation of the BP-MIB module. Each decision was made keeping in mind certain assumptions regarding the circumstances under which a DTN might commonly operate. These assumptions include:

o Low-powered equipment is likely to be used

o Limitations on long-term and (possibly) short-term storage are likely to exist and

o Delayed communication might require dissimilar management techniques from those used in timely environments

4.2.1. Node and Endpoint Representation

The first design decision involved the overall representation of Nodes and their respective attributes. According to section 3.1 of RFC 5050 [RFC5050], Nodes have a M:N relationship with Endpoints. That is to say, a Node can be a member of multiple Endpoints and (as long as it is not a Singleton Endpoint) an Endpoint can have multiple Nodes register it. Since SMIv2 [RFC2578] does not allow for the creation of three-dimensional conceptual tables, a normalized view of this relationship would be required; leading to the use of mediating tables. The method used here, as exhibited in section 5 of this document, was to emulate a three-dimensional conceptual table by having each instance of the BP-MIB contain managed information for a single Node - with each instance having a conceptual table containing managed information for each Endpoint of which it is a member.
4.2.2. Extension Blocks

Bundle Protocol Extension Blocks are mentioned in RFC 5050 [RFC5050], but are not defined there. The BP-MIB contains a set of Node-Specific objects that act as the list of Extensions Blocks that are available to the managed Node. These objects are arranged as a conceptual table that is indexed by the IANA defined Extension Block number (represented by bpExtBlockType). Also contained within the table is an OCTET STRING that acts as a human readable description of the table entry.

4.2.3. Convergence Layer Adapters

With the Node/Endpoint decision made, the next decision involved the representation of CLAs. Since multiple Nodes can share the use of CLAs, and since the characteristics of a CLA are implementation dependent, few objects were placed within the bpClaTable. Since it is likely that a CLA’s managed objects (those stored in a MIB designed specifically for the management of the given CLA type) will maintain more specific information than that contained in the bpClaTable, two objects were created as a way to reference a CLA’s MIB instance. These two objects are:

- bpClType - Identifies the type of Convergence Layer (CL) being used by the CLA (e.g. UDP, TCP, LTP, or otherwise)
- bpClID - Uniquely identifies the instance of managed information for the underlying CL whose type is defined by bpClType

The bpClType and bpClID objects are also used as a solution to a specific design goal. This goal involves the ability to trace managed information down the protocol stack and is discussed in more detail in section 4.2.4 of this document.

4.2.4. Managing the Protocol Stack

One of this MIB’s primary goals is to enable managed information to be traceable throughout the protocol stack. Ultimately, this goal is achieved by creating either a direct or indirect reference to an entry in the ifTable [RFC2863]. To make this possible, the BP-MIB MUST have identifier(s) that can be used to reference the MIB instance of the layer beneath itself. The BP-MIB is designed in such a way that both direct and indirect references to an ifTable entry can be achieved. This was done by placing two objects within the bpClaTable: one that uniquely identifies the CL type (bpClType) and one that uniquely identifies an instance of managed information for the given type (bpClID). Direct references to an ifTable entry are achieved by setting bpClType = 1 (which means it is referring to an
Indirect references are achieved when referencing a CLA MIB which, in turn, references the layer beneath itself.

bpClType SHOULD be a standardized numeric value that identifies the type of Convergence Layer being used by the Convergence Layer Adapter. This object is discussed more in section 4.2.5 of this document; giving rationale to the need for number standardization.

bpClID, alternatively, is an arbitrary value chosen by the agent that uniquely identifies an instance of the MIB whose type is given by bpClType. The NMS responsible for the managed information can use this value to determine the OID of the desired CLA.

Currently, the MIB module defines the bpClType object to be an integer. This design decision needs reconsideration however. For readability and ease of use, a textual-convention would likely be a better choice for the object syntax.

There are two objects in the bpClaTable that represent the number of Bundles passed into the BPA and out through the underlying CL (bpClInBndls and bpClOutBndls). Likewise, two additional objects exist to represent the amount of data passed between layers (bpClInBytes and bpClOutBytes). Though these values are not used for stack traversal, they are maintained as a way of offering a view of Bundle activity to/from the BP layer for each CLA.

4.2.5. bpClType Number Standardization

In sections 4.2.3 and 4.2.4 of this document, the use of bpClType was discussed. Using such an object, however, could result in inconsistencies across implementations of BP software. To avoid this, the BP-MIB uses a pre-defined set of numeric values, each uniquely representing a CL type.

4.2.6. Reason for bpClaDirection

The value in knowing the direction of data flow can be found in the use of certain equipment types. Spacecraft, for instance, are prone to utilizing an up-link that is different from the down-link. Scenarios similar to this are likely to result in the creation of two separate CLA MIB instances - one for each link. In these cases, a manager might desire knowing whether the underlying CL is capable of transmitting unidirectional traffic or bidirectional traffic. To represent this, the bpClaTable has been given the bpClaDirection object. The possible settings are inBound, outBound or biDirectional.
4.2.7. Notifications

In an attempt to keep things modular, TRAP and INFORM objects have been left out of the BP-MIB. These objects SHOULD be designed in a separate MIB module which, if used, MUST be implemented in conjunction with the BP-MIB. Since the requirements of network management within a DTN are distinct from those in a timely environment, use of notifications may vary between DTN environments and timely environments. With this in mind, careful consideration SHOULD be taken when designing such objects. This is discussed in more detail in section 4.3.3 of this document.

4.2.8. Statistical and Trended Data

Certain MIBs, such as those that manage high performance equipment, might contain mechanisms to store local data trends, simple statistical analysis, and bulk data counters (to avoid losing data when a counter resets to 0). The need for such mechanisms is likely to arise in certain DTN environments as well - but for different reasons. MIBs for high-performance equipment are likely to experience incredibly high counter iteration (essentially causing the counters to rollover before an NMS can retrieve the stored values) whereas a DTN is likely to accrue large amounts of data between NMS requests due to the constraints imposed by delay and disruption. However, in an attempt to keep things modular, statistic-based objects and trending objects have been considered beyond the scope of the BP-MIB. These objects, if any such objects are designed, SHOULD be put in a separate MIB module which MAY be implemented in conjunction with the BP-MIB, but can function independent of the BP-MIB. Design considerations for local trending and statistical analysis are addressed in section 4.3.1 of this document.

4.3. Considerations for Future Designers

This section is dedicated to describing certain considerations that future designers of DTN-related MIBs SHOULD observe in the design process. Section 4.3.1 gives a generic overview of DTN-related network management concepts and how they might apply to the design of DTN-related MIBs. Sections 4.3.2 through 4.3.4 observe particular MIB types and describes how they can and SHOULD inter-relate with the BP-MIB.

Though it’s beyond the scope of this document to discuss DTN network management requirements, certain DTN network management issues SHOULD be considered in the design of DTN-related MIBs. For this reason, this section contains a brief discussions that introduces potential road-blocks to future designers.
4.3.1. Delay/Disruption Entity Management

DTN-related MIB designers SHOULD consider the environment(s) under which the MIB might be deployed. Environment, as used here, refers to elements like: the storage capacity and processing power of the equipment being used; the frequency at which values might change in relation to the frequency at which they can be transmitted to an NMS; existing network management protocol capabilities and; whether best practices in timely networks apply or if distinctions ought to be made. Additionally, designers SHOULD consider the level of diversity that potential environments have from one another. Not doing so limits scalability and increases complexity for future designers.

In Delay / Disruption prone environments, high latency and periodic disconnection are likely to occur. With this in mind, designers SHOULD use object types that reflect an object’s nature - in regard to how frequently that object might change (e.g. how often a counter might roll back to 0). Designers SHOULD also consider that certain desirable statistics or trends might be impossible to calculate in the same way they could be in a timely environment. This might result in the need to calculate and store these values on the managed entity. Careful consideration SHOULD be given before using a MIB-based solution to this issue. Not doing so limits scalability and increases complexity for future designers.

4.3.2. CLA MIB Design

As described in sections 4.2.2 and 4.2.3, Convergence Layer Adapter MIBs will be needed for the management of information beyond what is offered by bpClaTable. This section addresses the design of those MIBs and offers suggestions to the designers.

Within the BP-MIB, the bpClaTable contains objects for each CLA the Node currently has access to. Within this table, two objects were defined as a reference to the underlying CLA’s managed objects. These objects (bpClType and bpClID) were described in detail in sections 4.2.3 and 4.2.4. This document RECOMMENDS that the design of CLA MIBs follow a similar pattern in order to preserve the goal of managing the protocol stack. To achieve this, two things MUST be done on the part of the designer. First, an object whose value can be referenced by bpClID needs to be placed somewhere within the CLA’s MIB. This object MUST be assigned an arbitrary, but unique, value for each instance of the object - however, the maintenance of uniqueness might be subject to software implementation rather than in the design of the MIB. Second, object(s) that can be used to reference the layer beneath the CLA need to be created. The exact method of doing this is beyond the scope of this document.
4.3.3. Notification MIB Design

Design of DTN-related TRAP and INFORM objects SHOULD take into account the environment(s) under which they might be deployed. The level of variance one environment’s conditions have to others’ can be arbitrarily small or overwhelmingly large. For instance, delay, in one environment, could refer to a relatively short time span, such as 5 minutes, whereas other environments might consider delay being in the order of days. TRAP and INFORM objects SHOULD NOT be designed as a solution to a single environment if they could reasonably be designed to encompass multiple environments.

With the introduction of delay into the management process, designers SHOULD consider whether notifications will be useful relative to the expected information arrival time. Since DTN elements can be managed both locally and remotely, designers of notifications SHOULD consider whether the notification can be used in both timely or delayed environments. To this same end, a designer SHOULD consider the ways (if any) notifications can be made to serve both environments.

4.3.4. Extension Block MIB Design

Extension blocks created for use with the Bundle Protocol are likely to contain characteristics not managed by the BP-MIB. However, the management of information regarding these characteristics SHOULD reference the Node by which the extension block was processed. There are two ways in which this MAY be done. First, a designer could reference the Node by its bpNodeID value. Second, since each Node is a member of at least one Singleton Endpoint, the Endpoint ID (EID) belonging to a Singleton of which the desired Node is a member can be used. This document however, RECOMMENDS use of bpNodeID. Use of a Singleton’s EID could fail in cases where a Node is a member of multiple Singleton Endpoints. As long as one other Singleton exists for the Node, it can unregister a Singleton Endpoint. If the EID used to reference the Node belonged to the unregistered Singleton, a management stations would no longer be able to know which Node is related to the extension block information.

5. Definitions

```
BP-MIB DEFINITIONS ::= BEGIN
IMPORTS
    enterprises, MODULE-IDENTITY, OBJECT-TYPE, Integer32,
    Gauge32, Counter64, TimeTicks
```
FROM SNMPv2-SMI
MODULE-COMPLIANCE, OBJECT-GROUP
FROM SNMPv2-CONF;

bpMIB MODULE-IDENTITY
LAST-UPDATED "201204191530Z"
ORGANIZATION "Ohio University IRG"
CONTACT-INFO
"Zack Sims
Phone: +1 (740)285-1895
Email: zack.sims@gmail.com"
"Hans Kruse
Phone: +1 (740) 593-4891
Email: kruse@ohio.edu
Alt: hkruse1@mac.com"

DESCRIPTION
"This MIB module was designed for the management of Bundle Protocol (BP) Nodes, for the Endpoints they are members of, and for Node <= CLA relationships. This MIB is structured in a way that Node-Specific objects are arranged separately from Endpoint and CLA-Specific objects. Objects which are Endpoint or CLA-Specific are arranged in Conceptual Rows within their own, respective, Conceptual Tables. Further, each instance of this module’s objects represents a single managed Node. This means that a separate instantiation of the BP-MIB’s objects will be forged for each Node being managed by the system. The layout of the MIB is arranged in the following order:

* Node-Specific Definitions
* Endpoint-Specific Definitions
* CLA-Specific Definitions
* Conformance Requirements for the BP-MIB"

REVISION "201204191530Z"
DESCRIPTION
"201103130330Z
The first public announcement of this MIB’s creation on the DTNRG (Delay Tolerant Networking Research Group) mailing list.

201105191930Z
The second release of the BP-MIB I-D (draft number 01). Changes include:
  * Adding CCSDS related objects (such as vrs#, Byte counts, and Extension Block tracking"
* Slight object renaming

* Addition of temporary textual conventions for IANA requirements.

201204191530Z
The third release of the BP-MIB I-D (draft number 02). Changes include:
* Clarity and consistency of description fields
* Object identity and purpose ===> sanity check
* Cross-standard compatibility check ===> CCSDS BP managed objects

-- These values are temporary until a DTN-MIB can get an IANA sanctioned value of its own ;)

::= { dtn 3 }
dtn OBJECT IDENTIFIER ::= { collegeOfComm  3 }
collegeOfComm OBJECT IDENTIFIER ::= { ohioUniv 42 }
ohioUniv OBJECT IDENTIFIER ::= { enterprises 32396 }

-- Node-Specific definitions

bpNodeObjects OBJECT IDENTIFIER ::= { bpMIB 1 }

bpNodeID OBJECT-TYPE
SYNTAX     OCTET STRING
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
  "An arbitrary value used to uniquely identify instances of the BP-MIB. This value can be used by other MIB definitions to reference instances of Bundle Node objects."
::= { bpNodeObjects 1 }

bpBundleVersion OBJECT-TYPE
SYNTAX     Integer32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
  "This value denotes which version number of the Bundle Protocol is being used by this Node."
::= { bpNodeObjects 2 }

bpLastUpTime OBJECT-TYPE
SYNTAX     TimeTicks
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"Is a representation of the last time this Node was
restarted and should be set by the system appropriately.
Some implementations of the Bundle Protocol might choose
to reset certain values when a Node is shutdown and brought
back up. This value is NOT a mirror of sysUpTime. Since
multiple Nodes can be running on a single system, this
value represents the last time this Node was reset."
::= { bpNodeObjects 3 }

bpAvailStorage OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This value represents the number of Kilo-Bytes currently
available to this Node for the storage of bundles. The value
ranges between 0 and (2^32) -1"
::= { bpNodeObjects 4 }

bpBulkPriorGenBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bundles generated on this
Node with a priority value of -bulk-

Note: the sum of bpNormPriorGenBndls, bpBulkPriorGenBndls,
and bpExpPriorGenBndls can be used to find the total of
Bundles generated at the local Node.

Bundle priority settings are mentioned in section 4.2 of RFC
5050"
::= { bpNodeObjects 5 }

bpBulkPriorGenBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bytes generated on this
Node with a priority value of -bulk-

Note: the sum of bpNormPriorGenBytes, bpBulkPriorGenBytes,
and bpExpPriorGenBytes can be used to find the total of
Bytes generated at the local Node.

Bundle priority settings are mentioned in section 4.2 of RFC 5050

::= { bpNodeObjects 6 }

bpNormPriorGenBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bundles generated on this Node with the priority value -normal-

Note: the sum of bpNormPriorGenBndls, bpBulkPriorGenBndls, and bpExpPriorGenBndls can be used to find the total of Bundles generated at the local Node.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050

::= { bpNodeObjects 7 }

bpNormPriorGenBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bytes generated on this Node with the priority value -normal-

Note: the sum of bpNormPriorGenBytes, bpBulkPriorGenBytes, and bpExpPriorGenBytes can be used to find the total of Bytes generated at the local Node.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050

::= { bpNodeObjects 8 }

bpExpPriorGenBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bundles generated on this Node with the priority value -expedited-

Note: the sum of bpNormPriorGenBndls, bpBulkPriorGenBndls, and bpExpPriorGenBndls can be used to find the total of
Bundles generated at the local Node.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050

::= { bpNodeObjects 9 }

bpExpPriorGenBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bytes generated on this Node with the priority value -expedited-

Note: the sum of bpNormPriorGenBytes, bpBulkPriorGenBytes, and bpExpPriorGenBytes can be used to find the total of Bytes generated at the local Node.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050

::= { bpNodeObjects 10 }

bpBulkPriorQueuedBndls OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bundles currently queued on this Node with the priority value -bulk-

Note: the sum of bpNormPriorGenBndls, bpBulkPriorGenBndls, and bpExpPriorGenBndls can be used to find the total of Bundles queued at the local Node.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050

::= { bpNodeObjects 11 }

bpBulkPriorQueuedBytes OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bytes currently queued on this Node with the priority value -bulk-

Note: the sum of bpNormPriorGenBytes, bpBulkPriorGenBytes, and bpExpPriorGenBytes can be used to find the total of
Bytes queued at the local Node.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050
::= { bpNodeObjects 12 }

bpNormPriorQueuedBndls OBJECT-TYPE
SYNTAX     Gauge32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"This keeps track of the number of Bundles currently queued on this Node with the priority value -normal-

Note: the sum of bpNormPriorGenBndls, bpBulkPriorGenBndls, and bpExpPriorGenBndls can be used to find the total of Bundles queued at the local Node.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050
::= { bpNodeObjects 13 }

bpNormPriorQueuedBytes OBJECT-TYPE
SYNTAX     Gauge32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"This keeps track of the number of Bytes currently queued on this Node with the priority value -normal-

Note: the sum of bpNormPriorGenBytes, bpBulkPriorGenBytes, and bpExpPriorGenBytes can be used to find the total of Bytes queued at the local Node.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050
::= { bpNodeObjects 14 }

bpExpPriorQueuedBndls OBJECT-TYPE
SYNTAX     Gauge32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"This keeps track of the number of Bundles currently queued on this Node with the priority value -expedited-

Note: the sum of bpNormPriorGenBndls, bpBulkPriorGenBndls, and bpExpPriorGenBndls can be used to find the total of
Bundles queued at the local Node.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050

::= { bpNodeObjects 15 }

bpExpPriorQueuedBytes OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bytes currently queued on this Node with the priority value -expedited-

Note: the sum of bpNormPriorGenBytes, bpBulkPriorGenBytes, and bpExpPriorGenBytes can be used to find the total of Bytes queued at the local Node.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050

::= { bpNodeObjects 16 }

bpReassemblyPendingBndls OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This value represents the current number of Bundles queued on this Node that have the retention constraint -Reassembly Pending-

RFC 5050 mentions the Reassembly Pending retention constraint in Sections 5.7 and 5.9. A retention constraint is any reason why a Bundle should not continue to be processed (whether it is waiting to reassemble fragments, dispatch, forward, or pass custody."

::= { bpNodeObjects 17 }

bpReassemblyPendingBytes OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This value represents the current number of Bytes queued on this Node that have the retention constraint -Reassembly Pending-

RFC 5050 mentions the Reassembly Pending retention
constraint in Sections 5.7 and 5.9. A retention constraint is any reason why a Bundle should not continue to be processed (whether it is waiting to reassemble fragments, dispatch, forward, or pass custody.

::= { bpNodeObjects 18 }

bpDispatchPendingBndls OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This value represents the current number of Bundles queued on this Node that have the retention constraint -Dispatch Pending-

RFC 5050 mentions the Dispatch Pending retention constraint in Sections 5.2, 5.4 and 5.6. A retention constraint is any reason why a Bundle should not continue to be (whether it is waiting to reassemble fragments, dispatch, forward, or pass custody."

::= { bpNodeObjects 19 }

bpDispatchPendingBytes OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This value represents the current number of Bytes queued on this Node that have the retention constraint -Dispatch Pending-

RFC 5050 mentions the Dispatch Pending retention constraint in Sections 5.2, 5.4 and 5.6. A retention constraint is any reason why a Bundle should not continue to be (whether it is waiting to reassemble fragments, dispatch, forward, or pass custody."

::= { bpNodeObjects 20 }

bpForwardPendingBndls OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This value represents the current number of Bundles queued on this Node that have the retention constraint -Forward Pending-

RFC 5050 mentions the Forward Pending retention constraint
in Sections 5.4 and 5.4.2. A retention constraint is any reason why a Bundle should not continue to be (whether it is waiting to reassemble fragments, dispatch, forward, or pass custody."

::= { bpNodeObjects 21 }

bpForwardPendingBytes OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This value represents the current number of Bytes queued on this Node that have the retention constraint -Forward Pending-

RFC 5050 mentions the Forward Pending retention constraint in Sections 5.4 and 5.4.2. A retention constraint is any reason why a Bundle should not continue to be (whether it is waiting to reassemble fragments, dispatch, forward, or pass custody."  

::= { bpNodeObjects 22 }

bpCustodyAcceptedBndls OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This value represents the current number of Bundles queued on this Node that have the retention constraint -Custody Accepted-

RFC 5050 mentions the Custody Accepted retention constraint in Sections 5.7 and 5.9. A retention constraint is any reason why a Bundle should not continue to be (whether it is waiting to reassemble fragments, dispatch, forward, or pass custody."  

::= { bpNodeObjects 23 }

bpCustodyAcceptedBytes OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This value represents the current number of Bytes queued on this Node that have the retention constraint -Custody Accepted-

RFC 5050 mentions the Custody Accepted retention constraint
in Sections 5.7 and 5.9. A retention constraint
is any reason why a Bundle should not continue to be
(whether it is waiting to reassemble fragments, dispatch,
forward, or pass custody.)"

```::= { bpNodeObjects 24 }
```

bpBundleReceptions OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A cumulative count of all the -Bundle Reception- reports
that this Node has received. Custody Acceptance reports are
received by the BPA in the event that a previously sent
Bundle requested it.

The Bundle status report request fields are listed in
Section 4.2 of RFC 5050. Each of the Bundle Status Reports
are defined in RFC 5050, Section 5.1 and are described in
greater detail in Section 6.1.1"

```::= { bpNodeObjects 25 }
```

bpCustodyAcceptance OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A cumulative count of all the -Custody Acceptance- reports
that this Node has received. Custody Acceptance reports are
received by the BPA in the event that a previously sent
Bundle requested it.

The Bundle status report request fields are listed in
Section 4.2 of RFC 5050. Each of the Bundle Status Reports
are defined in RFC 5050, Section 5.1 and are described in
greater detail in Section 6.1.1"

```::= { bpNodeObjects 26 }
```

bpBundleForwarding OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A cumulative count of all the -Bundle Forwarding- reports
that this Node has received. Bundle Forwarding reports are
received by the BPA in the event that a previously sent
Bundle requested it."
The Bundle status report request fields are listed in Section 4.2 of RFC 5050. Each of the Bundle Status Reports are defined in RFC 5050, Section 5.1 and are described in greater detail in Section 6.1.1

::= { bpNodeObjects 27 }

bpCustodySuccess OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A cumulative count of all the -Custody Success- reports that this Node has received. Custody Success reports are received by the BPA in the event that a previously sent Bundle requested it.

The Bundle status report request fields are listed in Section 4.2 of RFC 5050. Each of the Bundle Status Reports are defined in RFC 5050, Section 5.1 and are described in greater detail in Section 6.1.1

::= { bpNodeObjects 28 }

bpBundleDelivery OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A cumulative count of all the -Bundle Delivery- reports that this Node has received. Bundle Delivery reports are received by the BPA in the event that a previously sent Bundle requested it.

The Bundle status report request fields are listed in Section 4.2 of RFC 5050. Each of the Bundle Status Reports are defined in RFC 5050, Section 5.1 and are described in greater detail in Section 6.1.1

::= { bpNodeObjects 29 }

bpCustodyFailureBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of times this Node has failed to accept custody of incoming Bundles.

RFC 5050 defines custody transfer failures in Sections 5.10.1 and 5.12"
::= { bpNodeObjects 30 }

bpCustodyFailureBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of bytes this Node has failed to accept under its custody.

RFC 5050 defines custody transfer failures in Sections 5.10.1 and 5.12"

::= { bpNodeObjects 31 }

bpForwardFailureBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of bundles for which this Node has experienced a forwarding failure.

RFC 5050 defines forwarding failures in Sections 5.4.1 and 5.4.2"

::= { bpNodeObjects 32 }

bpForwardFailureBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of bytes for which this Node has experienced a forwarding failure.

RFC 5050 defines forwarding failures in Sections 5.4.1 and 5.4.2"

::= { bpNodeObjects 33 }

bpAbandonedDeliveryBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of bundles for which this Node has abandoned delivery.

RFC 5050 defines abandonment in Section 3.1"
bpAbandonedDeliveryBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of bytes for which this Node has abandoned
delivery.

    RFC 5050 defines abandonment in Section 3.1"
::= { bpNodeObjects 35 }

bpDiscardedBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of Bundles this Node has discarded.

    RFC 5050 defines Bundle discarding in Section 3.1"
::= { bpNodeObjects 36 }

bpDiscardedBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of Bytes this Node has discarded.

    RFC 5050 defines Bundle discarding in Section 3.1"
::= { bpNodeObjects 37 }

bpFragmentedLocally OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of Bundles that this Node has fragmented.

    RFC 5050 defines fragments in Section 3.1 and the
    fragmentation process in Section 5.8"
::= { bpNodeObjects 38 }

bpFragmentsCreated OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of Bundle Fragments that this Node has created
locally.

RFC 5050 defines fragments in Section 3.1 and the fragmentation process in Section 5.8
::= { bpNodeObjects 39 }

bpDelExpired OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of Bundles that this Node has deleted based on
-Lifetime Expired-

RFC 5050 defines the Bundle deletion process in Section 5.13
and the deletion status report flags in Section 6.1.1"
::= { bpNodeObjects 40 }

bpDelTransCancelled OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of Bundles that this Node has deleted based on
-Transmission Canceled-

RFC 5050 defines the Bundle deletion process in Section 5.13
and the deletion status report flags in Section 6.1.1"
::= { bpNodeObjects 41 }

bpDelDepStorage OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of Bundles that this Node has deleted based on
-Depleted Storage-

RFC 5050 defines the Bundle deletion process in Section 5.13
and the deletion status report flags in Section 6.1.1"
::= { bpNodeObjects 42 }

bpDelDestUnintel OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of Bundles that this Node has deleted based on
RFC 5050 defines the Bundle deletion process in Section 5.13 and the deletion status report flags in Section 6.1.1

::= { bpNodeObjects 43 }

bpDelNoRoute OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of Bundles that this Node has deleted based on
-No Known Route to Destination From Here-

::= { bpNodeObjects 44 }

bpDelUntimelyContact OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of Bundles that this Node has deleted based on
-No Timely Contact With Next Node on Route-

::= { bpNodeObjects 45 }

bpDelBlockUnintel OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of Bundles that this Node has deleted based on
-Block Unintelligible-

::= { bpNodeObjects 46 }

bpDelNoAdditionalInfo OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of Bundles that this Node has deleted based on
Note, certain software implementations might include extensions to the Bundle Protocol. Since extensions are prone to errors other than those specified by RFC 5050, this value may or may not increment upon extension-related errors (a counter of these errors may be contained in the extension’s MIB). It depends on the extension’s MIB design and the instrumentation of that MIB.

RFC 5050 defines the Bundle deletion process in Section 5.13 and the deletion status report flags in Section 6.1.1.

```plaintext
::= { bpNodeObjects 47 }

bpDelBytes OBJECT-TYPE
SYNTAX     Counter64
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"The total number of bytes deleted by this Node.

RFC 5050 defines the Bundle deletion process in Section 5.13 and the deletion status report flags in Section 6.1.1."
::= { bpNodeObjects 48 }

bpExtBlockTable OBJECT-TYPE
SYNTAX     SEQUENCE OF BPExtBlockEntry
MAX-ACCESS not-accessible
STATUS     current
DESCRIPTION
"A table whose rows identify the Bundle Protocol Extension Blocks available to this Node."
::= { bpNodeObjects 49 }

BPExtBlockEntry OBJECT-TYPE
SYNTAX     BPExtBlockEntry
MAX-ACCESS not-accessible
STATUS     current
DESCRIPTION
"This object represents a single entry in bpExtBlockTable. Each entry denotes the extension block’s type and offers a human readable description field."
INDEX
{ bpExtBlockType }
::= { bpExtBlockTable 1 }
bpExtBlockTypeOBJECT-TYPE
SYNTAX Integer32 (0..2147483647)
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"The value of this object uniquely identifies the type of
extension block being represented by this table entry. Each
extension block type is uniquely represented by a positive
integer.

Note, this object is currently in need of IANA
consideration and would most likely be better suited as a
textual-convention."
::= { bpExtBlockEntry 1 }

bpExtBlockDisplayName OBJECT-TYPE
SYNTAX OCTET STRING
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This is a human readable string value that stores either a
block name or short description of this this table entry."
::= { bpExtBlockEntry 2 }

-- Endpoint-Specific Definitions

bpRegObjects OBJECT IDENTIFIER ::= { bpMIB 2 }

bpRegCount OBJECT-TYPE
SYNTAX Gauge32
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This value does NOT keep track of the number of Endpoints
of which this Node is a member. Rather, it is directly
related to the number of rows in the bpRegTable. Because
Endpoints can be unregistered, this value may stay the same
while the number of Endpoint registrations decreases. This
value should never be set to 0 because every Node must be a
member of at least one Singleton Endpoint; see RFC 5050,
Section 3.1"
::= { bpRegObjects 1 }

bpRegTable OBJECT-TYPE
SYNTAX     SEQUENCE OF BPRegEntry
MAX-ACCESS not-accessible
STATUS     current
DESCRIPTION
"Is a sequence of Endpoint registrations and exists as a conceptual table for storing each registration entry and its managed objects. Each registration is uniquely identifiable by its EID (Endpoint ID). Each Node should be a member of at least one Singleton Endpoint, so this table should always have a minimum of 1 entry; see RFC 5050, Section 3.1"
::= { bpRegObjects 2 }

BPRegEntry OBJECT-TYPE
SYNTAX     BPRegEntry
MAX-ACCESS not-accessible
STATUS     current
DESCRIPTION
"A conceptual row of bpRegTable that contains the Endpoint-Specific objects for Endpoints of which this Node is a member. Each entry can be uniquely identified by its bpEndpointID. There should be at least one bpRegEntry in the bpRegTable at all times whose bpIsSingleton value is set to true; see RFC 5050, Section 3.1"
INDEX
{ bpRegIndex }
::= { bpRegTable 1 }

BPRegEntry ::= SEQUENCE
{
bpRegIndex                     Integer32,
bpEndpointID                   OCTET STRING,
bpCurState                     BITS,
bpIsSingleton                  BITS,
bpDeliveryFailureAction        BITS,
bpBundlesDeliveredBndls        Counter64,
bpBundlesDeliveredBytes        Counter64,
bpBundlesSentBndls             Counter64,
bpBundlesSentBytes             Counter64,
bpBundlesForwardedBndls        Counter64,
bpBundlesForwardedBytes        Counter64,
bpBundleDuplicatesBndls        Counter64,
bpBundleDuplicatesBytes        Counter64,
bpBulkPriorRecvdBndls          Counter64,
bpBulkPriorRecvdBytes          Counter64,
bpNormPriorRecvdBndls          Counter64,
bpNormPriorRecvdBytes          Counter64,
bpExpPriorRecvdBndls           Counter64,
bpExpPriorRecvdBytes           Counter64,
}
bpDelExpiredReports  Counter64,
bpDelTransCancelledReports  Counter64,
bpDelDepStorageReports  Counter64,
bpDelDestUnintelReports  Counter64,
bpDelNoRouteReports  Counter64,
bpDelUntimelyContactReports  Counter64,
bpDelBlockUnintelReports  Counter64,
bpDelNoAdditionalInfoReports  Counter64

bpRegIndex OBJECT-TYPE
SYNTAX     Integer32 (0..2147483647)
MAX-ACCESS not-accessible
STATUS     current
DESCRIPTION
"An index value that uniquely denotes a conceptual row of
the bpRegTable."
 ::= { bpRegEntry 1 }

bpEndpointID OBJECT-TYPE
SYNTAX     OCTET STRING
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"A string value representing the Endpoint ID of this
registered Endpoint.
Bundle Endpoints and Endpoint IDs are discussed in RFC 5050, Section 3.1"
 ::= { bpRegEntry 2 }

bpCurState OBJECT-TYPE
SYNTAX     BITS { passive(0), active(1) }
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"The current state of the BP (Bundle Protocol) on this
Endpoint.
Possible Values:
* passive - 0
* active - 1

The active and passive states of a registered Endpoint are
discussed in RFC 5050, Section 3.1"
 ::= { bpRegEntry 3 }
bpIsSingleton OBJECT-TYPE
SYNTAX     BITS { false(0), true(1) }
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"This value represents whether or not this registered
Endpoint is a Singleton.

Possible Values:
  * false  - 0 - Is not a Singleton Endpoint
  * true   - 1 - Is a Singleton Endpoint

Singleton Endpoints are discussed in RFC 5050, Section 3.1"
 ::= { bpRegEntry 4 }

bpDeliveryFailureAction OBJECT-TYPE
SYNTAX     BITS { abandonDelivery(0), deferDelivery(1) }
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"This value represents whether the registered Endpoint’s
Delivery Failure Action is set to defer or abandon.

Possible Values:
  * abandonDelivery - 0 - Abandon the Bundle’s delivery
  * deferDelivery   - 1 - Defer the Bundle’s Delivery

Bundle Delivery Failure Actions are discussed in RFC 5050,
Section 3.1"
 ::= { bpRegEntry 5 }

bpBundlesDeliveredBndls OBJECT-TYPE
SYNTAX     Counter64
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"A cumulative count of all the Bundles that have been
delivered to this Node on this Endpoint

Bundle delivery steps are discussed in RFC 5050, Section
5.7"
 ::= { bpRegEntry 6 }

bpBundlesDeliveredBytes OBJECT-TYPE
SYNTAX     Counter64
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"A cumulative count of all the Bytes that have been delivered to this Node on this Endpoint

Bundle delivery steps are discussed in RFC 5050, Section 5.7"
::= { bpRegEntry 7 }

bpBundlesSentBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A cumulative count of all the Bundles that have been sent from this Node through this Endpoint

Bundle reception and the steps to handling these Bundles are discussed in RFC 5050, Section 5.6"
::= { bpRegEntry 8 }

bpBundlesSentBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A cumulative count of all the Bytes that have been sent from this Node through this Endpoint

Bundle reception and the steps to handling these Bundles are discussed in RFC 5050, Section 5.6"
::= { bpRegEntry 9 }

bpBundlesForwardedBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A cumulative count of all the Bundles that have been forwarded to a Node, from this Node, that was a member of this Endpoint’s minimum reception group

The steps to handling Bundle forwarding are discussed in RFC 5050, Section 5.4"
::= { bpRegEntry 10 }

bpBundlesForwardedBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION

"A cumulative count of all the Bytes that have been
forwarded to a Node, from this Node, that was a member of
this Endpoint’s minimum reception group

The steps to handling Bundle forwarding are discussed in RFC
5050, Section 5.4"
::= { bpRegEntry 11 }

bpBundleDuplicatesBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION

"A cumulative count of all the duplicate Bundles that have
been received by this Node from this Endpoint. Duplicate
bundles are any bundle that arrives at a Node, but is
already within that Node’s storage."
::= { bpRegEntry 12 }

bpBundleDuplicatesBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION

"A cumulative count of Bytes that were received as
duplicate Bundles by this Node from this Endpoint.
Duplicate bundles are any bundle that arrives at a
Node, but is already within that Node’s storage."
::= { bpRegEntry 13 }

bpBulkPriorRecvdBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION

"This keeps track of the number of Bundles received on this
Node from this Endpoint with the priority value -bulk-

Note: the sum of bpNormPriorRecvdBndls,
bpBulkPriorRecvdBndls, and bpExpPriorRecvdBndls can be used
to find the total of Bundles that have been received by this
Node, from this Endpoint.

Bundle priority settings are mentioned in Section 4.2 of RFC
5050"
::= { bpRegEntry 14 }
bpBulkPriorRecvdBytes OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
 "This keeps track of the number of Bytes received on this Node from this Endpoint with the priority value -bulk-

Note: the sum of bpNormPriorRecvdBytes, bpBulkPriorRecvdBytes, and bpExpPriorRecvdBytes can be used to find the total of Bundles that have been received by this Node, from this Endpoint.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050"
 ::= { bpRegEntry 15 }

bpNormPriorRecvdBndls OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
 "This keeps track of the number of Bundles received on this Node from this Endpoint with the priority value -normal-

Note: the sum of bpNormPriorRecvdBndls, bpBulkPriorRecvdBndls, and bpExpPriorRecvdBndls can be used to find the total of Bundles that have been received by this Node, from this Endpoint.

Bundle priority settings are mentioned in Section 4.2 of RFC 5050"
 ::= { bpRegEntry 16 }

bpNormPriorRecvdBytes OBJECT-TYPE
SYNTAX      Counter64
MAX-ACCESS read-only
STATUS      current
DESCRIPTION
 "This keeps track of the number of Bytes received on this Node from this Endpoint with the priority value -normal-

Note: the sum of bpNormPriorRecvdBytes, bpBulkPriorRecvdBytes, and bpExpPriorRecvdBytes can be used to find the total of Bundles that have been received by this Node, from this Endpoint.

Bundle priority settings are mentioned in Section 4.2 of RFC
5050"
 ::= { bpRegEntry 17 }

bpExpPriorRecvdBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bundles received on this
Node from this Endpoint with the priority value -expedited-

Note: the sum of bpNormPriorRecvdBndls,
bpBulkPriorRecvdBndls, and bpExpPriorRecvdBndls can be used
to find the total of Bundles that have been received by this
Node, from this Endpoint.

Bundle priority settings are mentioned in Section 4.2 of RFC
5050"
 ::= { bpRegEntry 18 }

bpExpPriorRecvdBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"This keeps track of the number of Bytes received on this
Node from this Endpoint with the priority value -expedited-

Note: the sum of bpNormPriorRecvdBytes,
bpBulkPriorRecvdBytes, and bpExpPriorRecvdBytes can be used
to find the total of Bundles that have been received by this
Node, from this Endpoint.

Bundle priority settings are mentioned in Section 4.2 of RFC
5050"
 ::= { bpRegEntry 19 }

bpDelExpiredReports OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"The number of deletion reports received by this Node from
this Endpoint with the reason -Lifetime Expired-

RFC 5050 defines the bundle deletion process in Section 5.13
and the deletion status report flags in Section 6.1.1"
bpDelTransCancelledReports OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of deletion reports received by this Node from this Endpoint with the reason -Transmission Canceled-

RFC 5050 defines the bundle deletion process in Section 5.13 and the deletion status report flags in Section 6.1.1"
::= { bpRegEntry 21 }

bpDelDepStorageReports OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of deletion reports received by this Node from this Endpoint with the reason -Depleted Storage-

RFC 5050 defines the bundle deletion process in Section 5.13 and the deletion status report flags in Section 6.1.1"
::= { bpRegEntry 22 }

bpDelDestUninteligibleReports OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of deletion reports received by this Node from this Endpoint with the reason -Destination Endpoint ID Unintelligible-

RFC 5050 defines the bundle deletion process in Section 5.13 and the deletion status report flags in Section 6.1.1"
::= { bpRegEntry 23 }

bpDelNoRouteReports OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION "The number of deletion reports received by this Node from this Endpoint with the reason -No Known Route to Destination From Here-

RFC 5050 defines the bundle deletion process in Section 5.13 and the deletion status report flags in Section 6.1.1"
::= { bpRegEntry 24 }

bpDelUn timelyContactReports OBJECT-TYPE
SYNTAX     Counter64
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
 "The number of deletion reports received by this Node from
this Endpoint with the reason -No Timely Contact With Next
Node on Route-

RFC 5050 defines the bundle deletion process in Section 5.13
and the deletion status report flags in Section 6.1.1"
::= { bpRegEntry 25 }

bpDelBlockUnintelReports OBJECT-TYPE
SYNTAX     Counter64
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
 "The number of deletion reports received by this Node from
this Endpoint with the reason -Block Unintelligible-

RFC 5050 defines the bundle deletion process in Section 5.13
and the deletion status report flags in Section 6.1.1"
::= { bpRegEntry 26 }

bpDelNoAdditionalInfoReports OBJECT-TYPE
SYNTAX     Counter64
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
 "The number of deletion reports received by this Node from
this Endpoint for the reason -No Additional Information-

Note, certain software implementations might include
extensions to the Bundle Protocol. Since extensions are
capable of creating new status report types, this value may
or may not increment from errors outside the scope of RFC
5050. (another MIB might manage this information). It
depends on the extension’s MIB design and the
instrumentation of that MIB.

RFC 5050 defines the bundle deletion process in Section 5.13
and the deletion status report flags in Section 6.1.1"
::= { bpRegEntry 27 }

-- CLA-Specific Definitions
bpClObjects OBJECT IDENTIFIER ::= { bpMIB 3 }

bpClaCount OBJECT-TYPE
SYNTAX     Gauge32
MAX-ACCESS read-only
STATUS     current
DESCRIPTION "This is a value that keeps track of how many rows are currently in the bpClaTable. This value does NOT necessarily reflect the current number of CLAs (Convergence Layer Adapters) available to this Node. Since CLAs are implementation dependent, there are few qualifiers for a generic set of CLA-related manageable objects. A single Node can have 0 or more CLAs available, so this value can be set to 0.

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7"
::= {bpClObjects 1 }

bpClaTable OBJECT-TYPE
SYNTAX     SEQUENCE OF BPClaEntry
MAX-ACCESS not-accessible
STATUS     current
DESCRIPTION "This Conceptual table contains the set of CLAs (Convergence Layer Adapters) that are available to this Node. A CLA is an interface between the BPA (Bundle Protocol Agent) and an internetwork protocol suite. CLAs and their functions are implementation specific, so this set of managed information doesn’t contain much in the way of detail.

The items in this table are similar to the ifStackTable in RFC 2863, albeit different in that it only tracks the layer below the BPA (because the Bundle Protocol rests at the application layer and anything above itself would be done as a tunnel). Each row represents a single CLA; what Convergence Layer it’s using, and layer traversal information.

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7"
::= {bpClObjects 2 }

bpClaEntry OBJECT-TYPE
SYNTAX     BPClaEntry
MAX-ACCESS not-accessible
STATUS     current
DESCRIPTION

"Each entry represents a single CLA (Convergence Layer Adapter) of this Node. A CLA, as defined in RFC 5050, is an interface between the Bundle Protocol Agent and an inter-network protocol suite. Each bpClaEntry contains a reference to the layer beneath the BPA or, worded differently, to its CL (Convergence Layer). It’s the job of the bpClType object in each row to denote what type of protocol the CL is. If the underlying layer is not an inter-networking protocol, the bpClType is set to 1 (which signifies that it’s an interface).

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7"

INDEX
{ bpClaIndex }
::= { bpClaTable 1 }

BPClaEntry ::= SEQUENCE
{
  bpClaIndex                       Integer32,
  bpClType                         Integer32,
  bpClaDisplayName                 OCTET STRING,
  bpClID                           OCTET STRING,
  bpClOutBndls                     Counter64,
  bpClOutBytes                     Counter64,
  bpClInBndls                      Counter64,
  bpClInBytes                      Counter64,
  bpClaDirection                   BITS
}

bpClaIndex OBJECT-TYPE
SYNTAX     Integer32 (0..2147483647)
MAX-ACCESS not-accessible
STATUS     current
DESCRIPTION
  "An index value that is used to uniquely represent each conceptual row of the bpClaTable."
::= { bpClaEntry 1 }

bpClType OBJECT-TYPE
SYNTAX     Integer32 (0..2147483647)
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
  "With the desire to avoid dependence on any specific underlying protocols or protocol-suite, this value exists to specify the underlying CL (Convergence Layer) type being
used by this CLA (Convergence Layer Adapter). Each known CL type will be associated to a unique integer value greater than 0. Doing this will help to avoid implementation inconsistencies.

Each value herein uniquely represents a Convergence Layer type and, likewise, can be used in conjunction with the bpClID value to find managed information regarding an instance of the appropriate CLA’s MIB.

Convergence Layer Values:

<table>
<thead>
<tr>
<th>Interface</th>
<th>bpClID</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface</td>
<td>1</td>
</tr>
<tr>
<td>LTP</td>
<td>2</td>
</tr>
<tr>
<td>UDP</td>
<td>3</td>
</tr>
<tr>
<td>TCP</td>
<td>4</td>
</tr>
<tr>
<td>IP</td>
<td>5</td>
</tr>
<tr>
<td>DCCP</td>
<td>6</td>
</tr>
</tbody>
</table>

NOTE: if this convention becomes accepted, these values should probably make their way into a textual-convention rather than being loosely defined in a description block!

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7.

::= { bpClaEntry 2 }

bpClaDisplayName OBJECT-TYPE
SYNTAX OCTET STRING
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A string value that represents a locally assigned, human-readable name given to this CLA (Convergence Layer Adapter) by the BPA (Bundle Protocol Agent); assuming the BPA implementation has this capability.

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7.

::= { bpClaEntry 3 }

bpClID OBJECT-TYPE
SYNTAX OCTET STRING
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"An arbitrary value that uniquely identifies an instance of managed information for the CLA (Convergence Layer Adapter) which is associated to this entry.
Interpretation of this value is opaque to a user, but can be used by the management agent to determine which instance of a CLA’s MIB is referenced by this table entry. The format of this UID ought to be determined by implementors of BPA (Bundle Protocol Agent) software and it should be noted that no two conceptual rows in the bpClaTable (for a given Node) should have the same bpClID value.

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7
::= { bpClaEntry 4 }

bpClOutBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A counter that keeps track of the number of Bundles passed from the BPA (Bundle Protocol Agent) to the CL (Convergence Layer) through this CLA (Convergence Layer Adapter).

Note: CLA implementation compliance is limited in definition (see RFC 5050), and may result in many styles of CLA MIB implementation. Some may use a single CLA to handle inbound and outbound traffic. Other implementations may use one CLA for inbound traffic and a separate CLA for outbound. To know if this CLA is uni-directional or bidirectional, look at bpClaDirection.

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7
::= { bpClaEntry 5 }

bpClOutBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A counter that keeps track of the number of Bytes passed from the BPA (Bundle Protocol Agent) to the CL (Convergence Layer) through this CLA (Convergence Layer Adapter).

Note: CLA implementation compliance is limited in definition (see RFC 5050), and may result in many styles of CLA MIB implementation. Some may use a single CLA to handle inbound and outbound traffic. Other implementations may use one CLA for inbound traffic and a separate CLA for outbound. To know if this CLA is uni-directional or bidirectional, look at
bpClaDirection.

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7
 ::= { bpClaEntry 6 }

bpClInBndls OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A counter that keeps track of the number of Bundles passed to the BPA (Bundle Protocol Agent) from the CL (Convergence Layer) through this CLA (Convergence Layer Adapter).

Note: CLA implementation compliance is limited in definition (see RFC 5050), and may result in many styles of CLA MIB implementation. Some may use a single CLA to handle inbound and outbound traffic. Other implementations may use one CLA for inbound traffic and a separate CLA for outbound. To know if this CLA is uni-directional or bidirectional, look at bpClaDirection.

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7"
 ::= { bpClaEntry 7 }

bpClInBytes OBJECT-TYPE
SYNTAX Counter64
MAX-ACCESS read-only
STATUS current
DESCRIPTION
"A counter that keeps track of the number of Bytes passed to the BPA (Bundle Protocol Agent) from the CL (Convergence Layer) through this CLA (Convergence Layer Adapter).

Note: CLA implementation compliance is limited in definition (see RFC 5050), and may result in many styles of CLA MIB implementation. Some may use a single CLA to handle inbound and outbound traffic. Other implementations may use one CLA for inbound traffic and a separate CLA for outbound. To know if this CLA is uni-directional or bidirectional, look at bpClaDirection.

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7"
 ::= { bpClaEntry 8 }
bpClaDirection OBJECT-TYPE
SYNTAX     BITS { inBound(0), outBound(1), biDirectional(2) }
MAX-ACCESS read-only
STATUS     current
DESCRIPTION
"A value representing the nature of the CLA (Convergence Layer Adapter) and whether the BPA (Bundle Protocol Agent) can send/receive to/from the CL through the CLA. Some implementations of BP software, such as satellite software, may find it preferable to have uni-directional traffic while others may prefer to have a single entry resolve all traffic (bi-directional).

Possible Values:
* inBound       - 0 - inbound traffic only
* outBound      - 1 - outbound traffic only
* biDirectional - 2 - bi-directional traffic

CLAs are described in RFC 5050 in Section 3.1 and CLs are discussed in Section 7"
::= { bpClaEntry 9 }

-- Conformance Requirements for the BP-MIB

bpConformance OBJECT IDENTIFIER ::= { bpMIB 4 }
bpMIBCompliance OBJECT IDENTIFIER ::= { bpConformance 1 }

bpCompliance MODULE-COMPLIANCE
STATUS     current
DESCRIPTION
"The compliance for Network Elements that are running a Bundle Protocol Implementation."
MODULE
GROUP       bpNodeInfoGroup
DESCRIPTION
"Node-Specific information regarding the Node itself and the Bundles/Bytes which are currently queued by it."
GROUP       bpNodeActivityGroup
DESCRIPTION
"Node-Specific information that offers a view of what a Node has done locally and what sorts of actions or events the Node may have undergone."
GROUP       bpNodeDelGroup
DESCRIPTION
"Node-Specific information that keeps counters for the different reasons a Bundle has been deleted on this Node. It also tracks a cumulative count of Bytes that have been on this Node."
GROUP       bpRegInfoGroup
DESCRIPTION  "Endpoint-Specific information directly regarding the Node’s
Endpoint registrations. Things like: EID, state, Singleton
status, and delivery failure action."
GROUP       bpRegActivityGroup
DESCRIPTION  "Endpoint-Specific information that keeps counters for the
amount of Bundles/Bytes that fit a certain criterion, such
as: sends, receives, local deliveries, etc"
GROUP       bpRegDelStatusGroup
DESCRIPTION  "Endpoint-Specific information that keeps counters for the
number of status reports received on this Endpoint for this
Node."
GROUP       bpClaGroup
DESCRIPTION  "CLA-Specific Information that relates to a Bundle Node’s
Convergence Layer Adapters and the corresponding Convergence
Layers themselves"
::= { bpMIBCompliance 1 }

-- BP MIB Group Definitions

bpMIBGroups OBJECT IDENTIFIER ::= { bpConformance 2 }

bpNodeInfoGroup OBJECT-GROUP
OBJECTS
{
  bpNodeID,
  bpBundleVersion,
  bpAvailStorage,
  bpLastUpTime,
  bpBulkPriorQueuedBndls,
  bpBulkPriorQueuedBytes,
  bpNormPriorQueuedBndls,
  bpNormPriorQueuedBytes,
  bpExpPriorQueuedBndls,
  bpExpPriorQueuedBytes,
  bpReassemblyPendingBndls,
  bpReassemblyPendingBytes,
  bpDispatchPendingBndls,
  bpDispatchPendingBytes,
  bpForwardPendingBndls,
  bpForwardPendingBytes,
  bpCustodyAcceptedBndls,
  bpCustodyAcceptedBytes,
  bpExtBlockDisplayName
bpNodeActivityGroup OBJECT-GROUP
OBJECTS
{
  bpBulkPriorGenBndls,
  bpBulkPriorGenBytes,
  bpNormPriorGenBndls,
  bpNormPriorGenBytes,
  bpExpPriorGenBndls,
  bpExpPriorGenBytes,
  bpCustodyFailureBndls,
  bpCustodyFailureBytes,
  bpForwardFailureBndls,
  bpForwardFailureBytes,
  bpAbandonedDeliveryBndls,
  bpAbandonedDeliveryBytes,
  bpDiscardedBndls,
  bpDiscardedBytes,
  bpFragmentedLocally,
  bpFragmentsCreated,
  bpBundleReceptions,
  bpCustodyAcceptance,
  bpBundleForwarding,
  bpCustodySuccess,
  bpBundleDelivery
}

bpNodeDelGroup OBJECT-GROUP
OBJECTS
{
  bpDelExpired,
  bpDelTransCancelled,
  bpDelDepStorage,
  bpDelDestUnintel,
  bpDelNoRoute,
  bpDelUntimelyContact,
bpDelBlockUnintelpDelNoAdditionalInfo, bpDelBytes

}  
STATUS current  
DESCRIPTION  
"Node-Specific information that keeps counters for the different reasons a Bundle has been deleted on this Node. It also tracks a cumulative count of Bytes that have been on this Node."

::= { bpMIBGroups 3 }

bpRegInfoGroup OBJECT-GROUP
OBJECTS
{
  bpRegCount, bpEndpointID, bpCurState, bpIsSingleton, bpDeliveryFailureAction
}
STATUS current  
DESCRIPTION  
"Endpoint-Specific information directly regarding the Node’s Endpoint registrations. Things like: EID, state, Singleton status, and delivery failure action."

::= { bpMIBGroups 4 }

bpRegActivityGroup OBJECT-GROUP
OBJECTS
{
}
STATUS current  
DESCRIPTION
"Endpoint-Specific information that keeps counters for the amount of Bundles/Bytes that fit a certain criterion, such as: sends, receives, local deliveries, etc"
::= { bpMIBGroups 5 }

bpRegDelStatusGroup OBJECT-GROUP
OBJECTS
{
  bpDelExpiredReports,
  bpDelTransCancelledReports,
  bpDelDepStorageReports,
  bpDelDestUnintelReports,
  bpDelNoRouteReports,
  bpDelUn timelyContactReports,
  bpDelBlockUnintelReports,
  bpDelNoAdditionalInfoReports
}
STATUS current
DESCRIPTION
"Endpoint-Specific information that keeps counters for the number of status reports received on this Endpoint for this Node."
::= { bpMIBGroups 6 }

bpClaGroup OBJECT-GROUP
OBJECTS
{
  bpClaCount,
  bpClType,
  bpClaDisplayName,
  bpClID,
  bpClOutBndls,
  bpClOutBytes,
  bpClInBndls,
  bpClInBytes,
  bpClaDirection
}
STATUS current
DESCRIPTION
"CLA-Specific Information that relates to a Bundle Node’s Convergence Layer Adapters and the corresponding Convergence Layers themselves"
::= { bpMIBGroups 7 }

END
6. Acknowledgments

Extensive thanks is given to Hans Kruse, Dovel Myers, Gilbert Clark, Josh Schendel, and the family of Ohio University’s Internetworking Research Group Laboratory for their contributions on the MIB’s design and how it might inter-relate with other DTN-related MIBs.

Additional thanks to Scott Burleigh of NASA Jet Propulsion Laboratory for asking relative design questions and for offering helpful implementation suggestions.

7. IANA Considerations

This section has work to be done. IANA considerations do include: the standardization of numbers to uniquely identify Convergence Layer types; creation of a Textual-Convention for bpClType; and creation of a Textual-Convention for bpExtBlockType.

8. Security Considerations

Though SNMPv3 is the current de facto standard of network management protocols, there is no guarantee that this will hold true in the DTN architecture. With this in mind, any protocols used SHOULD consider, to the fullest feasible extent, the security measures used in SNMPv3. Design of such protocols SHOULD consider the lessons learned in the terrestrial Internet. The following paragraphs address issues specific to SNMP, but may still hold relevance for DTN network management mechanisms.

There are no management objects defined in this MIB module that have a MAX-ACCESS clause of read-write and/or read-create. So, if this MIB module is implemented correctly, then there is no risk that an intruder can alter or create any management objects of this MIB module via direct SNMP SET operations.

SNMP versions prior to SNMPv3 did not include adequate security. 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 module.

It is RECOMMENDED that implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410], section 8), including full support for the SNMPv3 cryptographic mechanisms (for authentication and privacy).

Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to
enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module 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.

9. References

9.1. Normative References


9.2. Informative References

Authors’ Addresses

Zack A. Sims
Ohio University

Phone: +1 740 285 1895
EMail: zack.sims@gmail.com

Hans Kruse
Ohio University
292 Lindley Hall, Ohio University
Athens, Ohio 45662
United States

Phone: +1 740 593 4891
EMail: kruse@ohio.edu