Emerging Internet Management Technologies

Jürgen Schönwälder

Computer Science Department
Technical University Braunschweig
Bültenweg 74/75
38106 Braunschweig
Germany

Phone: +1 49 531 391 3289
Email: `schoenw@ibr.cs.tu-bs.de`
Web: `http://www.ibr.cs.tu-bs.de/~schoenw`

Outline of the Tutorial

1. Simple Network Management Protocol Version 3	90 Minutes
2. Agent Extensibility Protocol	50 Minutes
3. Distributed Internet Management	50 Minutes
4. Summary and Perspective	20 Minutes

1. Simple Network Management Protocol Version 3

1.1 Historic Perspective

1.2 Architectural Concepts

1.3 Message Processing and Protocol Operations

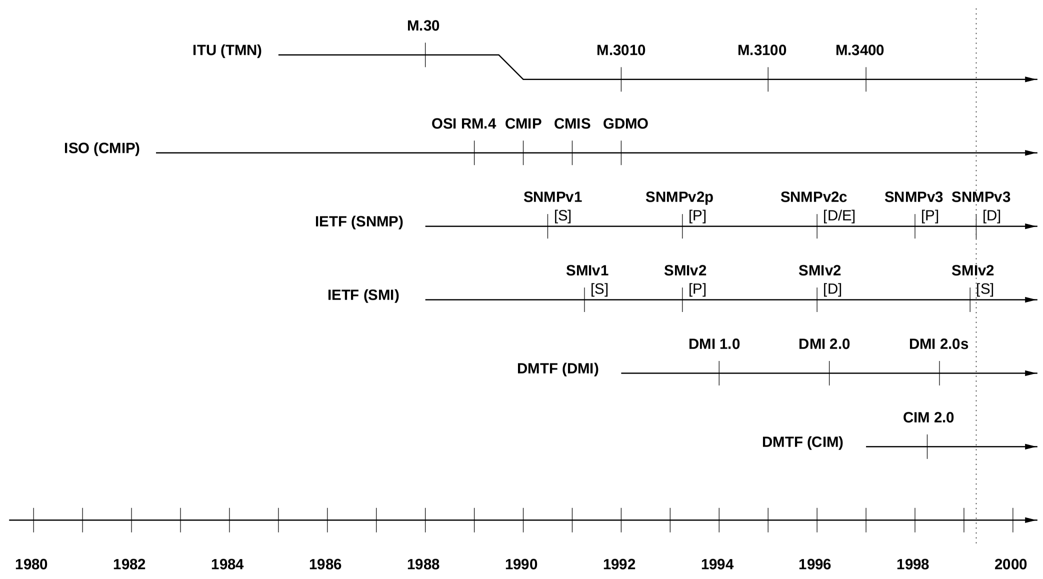
1.4 Message Security and Access Control

1.5 SNMP Applications

1.6 Coexistence with SNMPv1/SNMPv2c

1.7 Implementations, Products, Experiences

History of Management Standards



SNMP History and RFC Overview

- “SNMP” (SNMPv1) circa 1988-1991 [RFC 1155, 1157, 1212, 1213, 1215]
- “SNMP Security” and “SMP” circa 1991-1992
- “Party-based SNMPv2” (SNMPv2p) circa 1993-1995 [RFC 1441–1452]
- “Community-based SNMPv2” (SNMPv2c) circa 1996 [RFC 1901–1907]
- “SNMPv2u” and “SNMPv2*” circa 1996
- “SNMPv3” circa 1997-1999 [RFC 1905–1907, 2271–2275, 2578–2580]

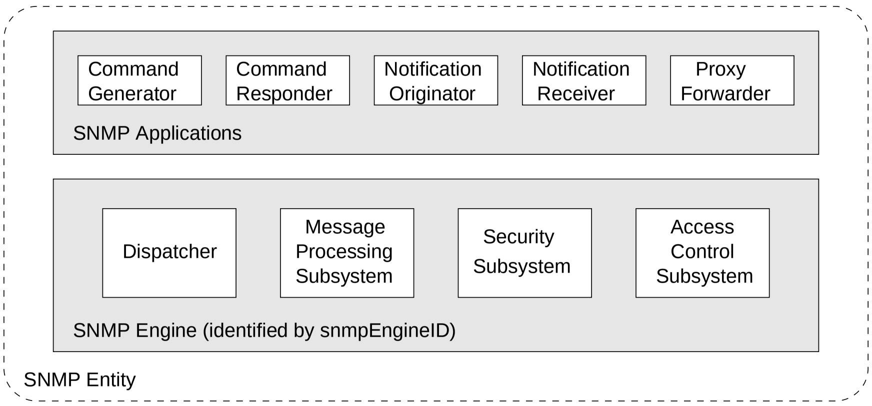
SNMPv3 Framework

- Data Definition Language (SMI)
  - SMIv2 defined in RFC 2578–2580 (STD 58)
- Definitions of Management Information (MIBs)
  - Nearly 100 IETF MIB modules containing roughly 10.000 definitions.
  - Even larger and growing number of enterprise-specific MIB modules.
- Protocol Operations and Transport Mappings
  - RFC 1905–1907 (Draft Standard)
  - Currently under revision for full Internet Standard status.
- Security and Administration
  - RFC 2271–2275 (Proposed Standard)
  - Publication of revised versions as Draft Standard expected in May/June.

Architectural Goals of SNMPv3

- Address the need for secure SNMP (write) operations.
- Define an architecture that allows for longevity of SNMP frameworks.
- Support inexpensive minimal conforming implementations.
- Support more complex conforming implementations required in large networks.
- Allow to move portions of the architecture along the IETF standards track.
- Use existing materials as much as possible.
- Keep SNMP as simple as possible.

SNMP Entities

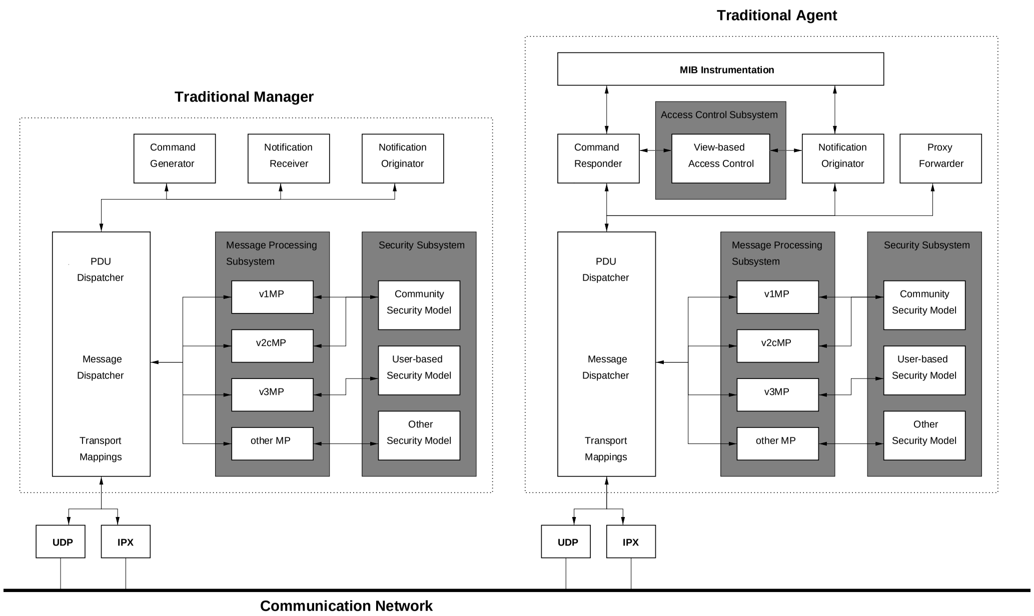


- Fine grained SNMP applications instead of coarse grained agents and managers.
- Exactly on engine per SNMP entity and exactly one dispatcher per SNMP engine.
- Every abstract subsystem may consist of one or more concrete models.
- Modularization enables incremental enhancements to SNMP.

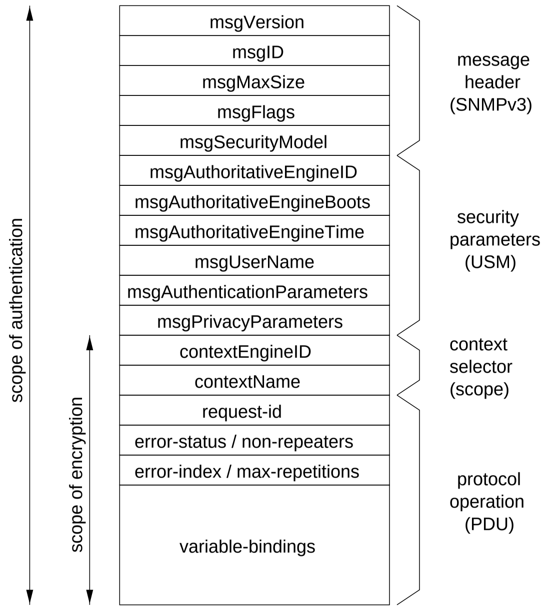
SNMP Contexts

- An context is a collection of management information accessible by an SNMP entity.
  - An SNMP entity potentially has access to many contexts.
  - An item of management information may exist in more than one context.
- Within a management domain, a managed object is uniquely identified by:
  1. the identification of the engine within the SNMP entity (e.g. “xyz”)
  2. the context name within the SNMP entity (e.g. “board1”)
  3. the managed object type (e.g. “IF-MIB.ifDescr”)
  4. the instance identifier (e.g. “1”)

Manager and Agent in the SNMP Architecture

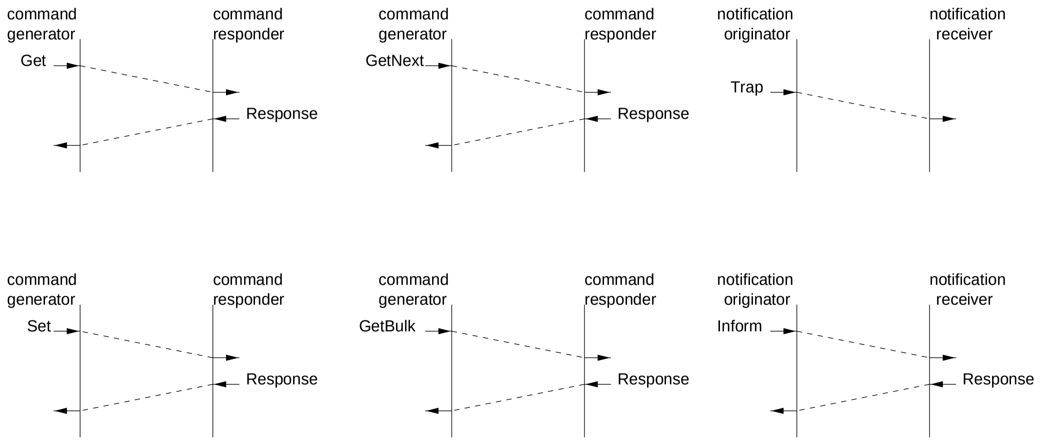


SNMPv3/USM Messages (RFC 2272, 2274)



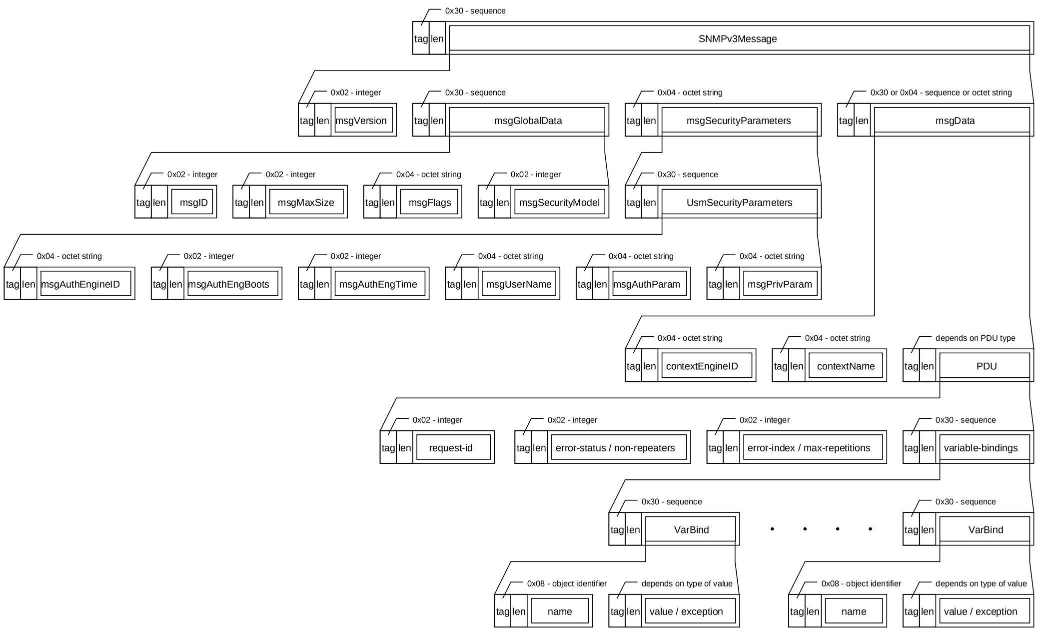
- `msgVersion` identifies the message processing model.
- `msgSecurityModel` identifies the security model.
- `contextEngineID` and `contextName` determine the context.
- The type of protocol operation (and version) is determined by the tag of the PDU.

SNMPv3 Protocol Operations (RFC 1905)



- An additional `Report` protocol operation is used internally for error notifications, engine discovery and clock synchronization.

Encoding of SNMPv3/USM Messages



Classes of Protocol Operations

- The processing of a message depends on the class of the embedded protocol operation:

Class	Description
Read	PDUs that retrieve management information.
Write	PDUs which attempt to modify management information.
Response	PDUs which are sent in response to a request.
Notification	PDUs which transmit event notifications.
Internal	PDUs exchanged internally between SNMP engines.
Confirmed	PDUs which cause the receiver to send a response.
Unconfirmed	PDUs which are not acknowledged.

- The introduction of PDU classes enables the IETF to add new protocol operations without having to update the message processing specification.

- There is no explicit support in the message format to indicate the protocol operations supported/used by an SNMP engine.

SNMPv3 Error and Exception Handling

SNMPv3 Exception	Get	GetNext/GetBulk	SNMPv1 Error Status
noSuchObject	X		noSuchName(2)
noSuchInstance	X		noSuchName(2)
endOfMibView		X	noSuchName(2)

- Error handling in SNMPv1:
  - An error response contains an error status and an error index.
  - Error responses contain no useful management information.
  - There is only a single error status and error index even if there are multiple errors.
- Error and exception handling in SNMPv3:
  - Per variable-binding exceptions in common error situations.
  - One or more exceptions are not considered to be an error condition.
  - A response with exceptions still contains useful management information.
  - Other errors are handled as in SNMPv1 with more detailed error status codes.

⇒ An SNMPv3 command generator must be prepared to deal with SNMPv1 error responses that may come from proxied SNMPv1 command responders.

SNMPv3 and SNMPv1 Error Codes

SNMPv3 Error Code	Read Class	Write Class	Notification Class	SNMPv1 Error Code
noError(0)	X	X	X	noError(0)
tooBig(1)	X	X	X	tooBig(1)
noSuchName(2)				noSuchName(2)
badValue(3)				badValue(3)
readOnly(4)				readOnly(4)
genErr(5)	X	X	X	genErr(5)
noAccess(6)		X		noSuchName(2)
wrongType(7)		X		badValue(3)
wrongLength(8)		X		badValue(3)
wrongEncoding(9)		X		badValue(3)
wrongValue(10)		X		badValue(3)
noCreation(11)		X		noSuchName(2)
inconsistentValue(12)		X		badValue(3)
resourceUnavailable(13)		X		genErr(5)
commitFailed(14)		X		genErr(5)
undoFailed(15)		X		genErr(5)
authorizationError(16)	X	X	X	noSuchName(2)
notWritable(17)		X		noSuchName(2)
inconsistentName(18)		X		noSuchName(2)

SNMPv3/USM Textual Conventions

- SnmpEngineID
  - Unique identification of an SNMP engine within a management domain.
- SnmpSecurityModel
  - Identification of a specific security model.
- SnmpMessageProcessingModel
  - Identification of a specific message processing model.
  - The message processing model is encoded in the `msgVersion`.
- SnmpSecurityLevel
  - The security level of a given message (`noAuthNoPriv`, `authNoPriv`, `authPriv`).
  - The security level is encoded in the `msgFlags`.
- KeyChange
  - Defines a cryptographic algorithm to change authentication or encryption keys.
  - Does not require encryption. An attacker can “drill forward” once a key is broken.

Security Issues

- The following questions must be answered in order to decide whether an operation should be performed or not:

  1. Is the message specifying an operation authentic?
  2. Who requested the operation to be performed?
  3. What objects are accessed in the operation?
  4. What are the rights of the requester with regard to the objects of the operation?

- 1 and 2 are answered by message security mechanisms (authentication and privacy).
- 3 and 4 are answered by authorization mechanisms (access control).

USM Message Security (RFC 2274)

- Protection against the following threads:
  1. Modification of Information
     (Unauthorized modification of in-transit SNMP messages.)
  2. Masquerade
     (Unauthorized users attempting to use the identity of authorized users.)
  3. Disclosure
     (Protection against eavesdropping on the exchanges between SNMP entities.)
  4. Message Stream Modification
     (Re-ordered, delayed or replayed messages to effect unauthorized operations.)

- No protection against:
  - Denial of Service
    (Denial of service attacks are usually indistinguishable from network failures.)
  - Traffic Analysis
    (No significant advantage afforded by protecting against traffic analysis.)

USM Security Services (RFC 2274)

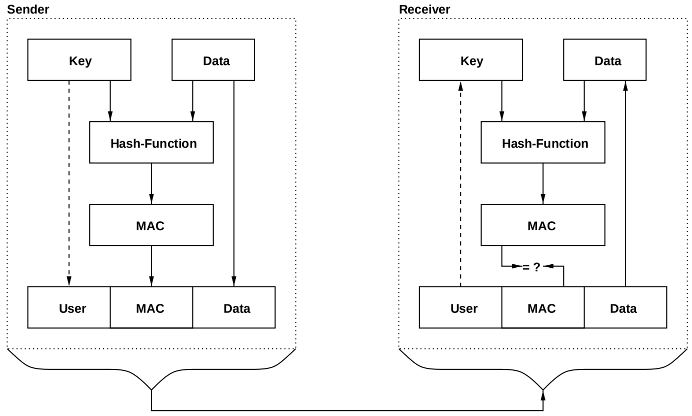
- Data Integrity
  - Data has not been altered or destroyed in an unauthorized manner.
  - Data sequences have not been altered to an extent greater than can occur non-maliciously.

- Data Origin Authentication
  - The claimed identity of the user on whose behalf received data was originated is corroborated.

- Data Confidentiality
  - Information is not made available or disclosed to unauthorized individuals, entities, or processes.

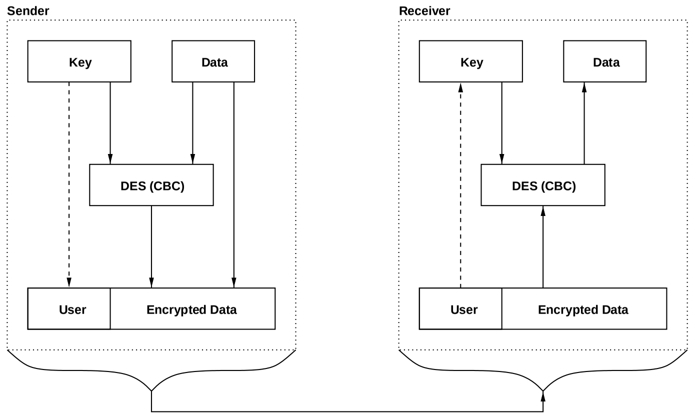
- Message Timeliness and Limited Replay Protection
  - A message whose generation time is outside of a time window is not accepted.
  - Message reordering is not dealt with and can occur in normal conditions too.

USM Data Integrity and Data Origin Authentication



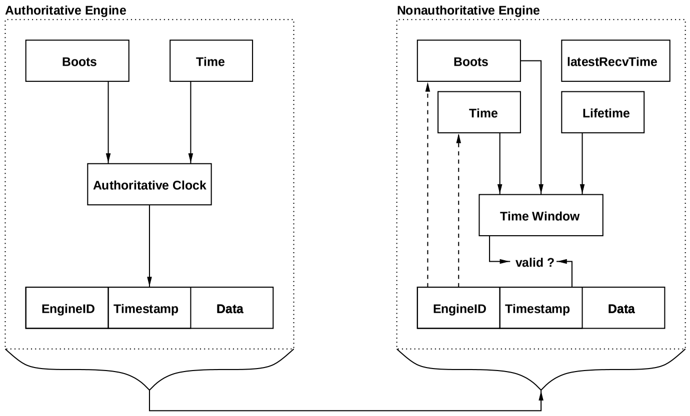
- Cryptographic strong hash functions generate a message authentication code (MAC).
- The MAC ensures integrity while the symmetric key provides for authentication.
- USM currently defines HMAC-MD5-96 and HMAC-SHA-96 (96 bits in the message).
- Other hash functions may be added in the future without changes to USM.

USM Data Confidentiality



- Optional encryption of the `ScopedPDU` using symmetric keys.
- USM currently defines CBC-DES (cipher-block-chaining mode).
- Other encryption functions may be added in the future without changes to USM.
- Encryption is relatively expensive and should only be used when needed.

USM Message Timeliness and Replay Protection



- A non-authoritative engine maintains a notion of the time at the authoritative engine.
- A non-authoritative engine also keeps track when the last authentic message was received from a given engine.
- A message is accepted and considered “fresh” if it falls within a time window.
- The maximum width of the time window is defined to be 300 seconds (5 minutes).

USM Passwords and Localized Keys

- Generating keys from passwords:
  - Algorithmic transformation of a human readable password into a cryptographic key.
  - Produce a string S of length $2^{20} = 1048576$ bytes by repeating the password as many times as necessary.
  - Compute the users key K_U using either $K_U = MD5(S)$ or $K_U = SHA(S)$.
  - Intended to slow down brute force password attacks.
  - No serious barrier for an attacker with a transformed dictionary.
- Generating localized keys:
  - Algorithmic transformation of the users key K_U and an engine identification E into a localized key.
  - For a given engine E, compute either $K_{UL} = MD5(K_U, E, K_U)$ or $K_{UL} = SHA(K_U, E, K_U)$.
  - Advantage: A compromised key does not give access to other SNMP engines.

USM Key Changes

- Key changes must be possible without encryption since encryption is optional.
- Key change procedure (initiator):
  1. Generate a random value r from a random number generator.
  2. Compute $d = MD5(K_{old}, r)$ or $d = SHA(K_{old}, r)$.
  3. Compute $\delta = d \oplus K_{new}$ and send (δ, r).
- Key change procedure (receiver):
  1. Compute $d = MD5(K_{old}, r)$ or $d = SHA(K_{old}, r)$.
  2. Compute $K_{new} = d \oplus \delta$.
- The receiver computes the correct new key since $d \oplus \delta = d \oplus (d \oplus K_{new}) = K_{new}$.
- An attacker who is able to catch all key updates can calculate the new keys once the old key is broken.

⇒ Use encryption for key changes if at all possible!

Authoritative Engine and Timeliness Checks

- Authoritative Engine:
  - Either the sender or the receiver of a message is designated the authoritative engine.
  - The receiver is authoritative if the message contains a confirmed class PDU.
  - The sender is authoritative if the message contains an unconfirmed class PDU.
- Timeliness check of an authoritative receiver:
  - A message is outside the time window if any of the following is true:
    1. `snmpEngineBoots` $= 2^{31} - 1$
    2. `msgAuthoritativeEngineBoots` $\neq$ `snmpEngineBoots`
    3. $abs($`msgAuthoritativeEngineTime` $-$ `snmpEngineTime`$) > 150$ seconds
- Timeliness check of a non-authoritative receiver:
  - A message is outside the time window if any of the following is true:
    1. `snmpEngineBoots` $= 2^{31} - 1$
    2. `msgAuthoritativeEngineBoots` $<$ `snmpEngineBoots`
    3. `msgAuthoritativeEngineBoots` $=$ `snmpEngineBoots` and `msgAuthoritativeEngineTime` $<$ `snmpEngineTime` -150

USM Clock Synchronization and Discovery

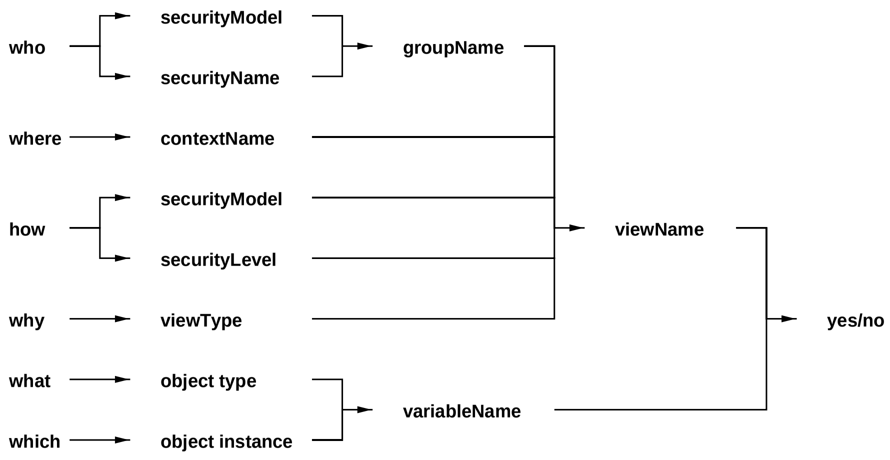
- An SNMP engine must remain loosely synchronized with other authoritative engines:
  - For each remote authoritative SNMP engine, an SNMP engine maintains: `snmpEngineBoots`, `snmpEngineTime` and `latestReceivedEngineTime`
  - Time synchronization only occurs if the message is authentic.
  - An update occurs, if at least one of the following conditions is true:
    1. `msgAuthoritativeEngineBoots` > `snmpEngineBoots`
    2. `msgAuthoritativeEngineBoots` = `snmpEngineBoots` and `msgAuthoritativeEngineTime` > `latestReceivedEngineTime`
- Discovery and initial clock synchronization:
  - The engine identification is needed to compute localized keys and to keep clock information for authoritative engines.
  - An SNMP engine can learn the engine identification by sending a `noAuthNoPriv` request with a zero-length `msgAuthoritativeEngineID`.
  - The receiver returns a `Report` PDU with the real `msgAuthoritativeEngineID`.
  - Similarly, (initial) clock synchronization happens by sending an authentic request and receiving a `Report` PDU with the authoritative time.

USM MIB (RFC 2274)

usmUserEntry
usmUserEngineID SnmpEngineID
usmUserName SnmpAdminString
usmUserSecurityName SnmpAdminString
usmUserCloneFrom RowPointer
usmUserAuthProtocol AutonomousType
usmUserAuthKeyChange KeyChange
usmUserOwnAuthKeyChange KeyChange
usmUserPrivProtocol AutonomousType
usmUserPrivKeyChange KeyChange
usmUserOwnPrivKeyChange KeyChange
usmUserPublic OctetString
usmUserStorageType StorageType
usmUserStatus RowStatus

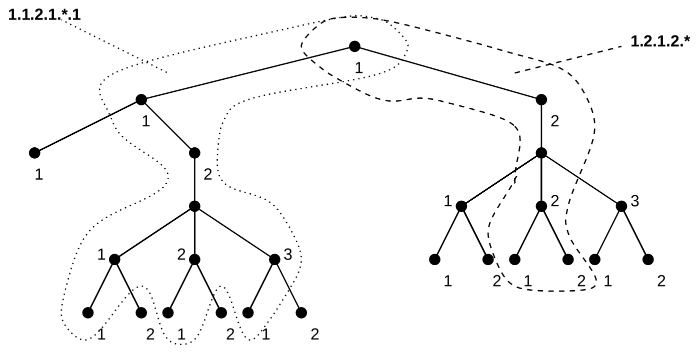
- The `usmUserTable` maps USM user names to `securityNames`.
- New entries may be created by cloning existing entries together with their keys.
- The `usmUserAuthKeyChange` and `usmUserPrivKeyChange` objects may be used by the security administrator to change the user's keys.
- The `usmUserOwnAuthKeyChange` and `usmUserOwnPrivKeyChange` objects may be used by the user to change his keys.

View-based Access Control Logic (RFC 2275)



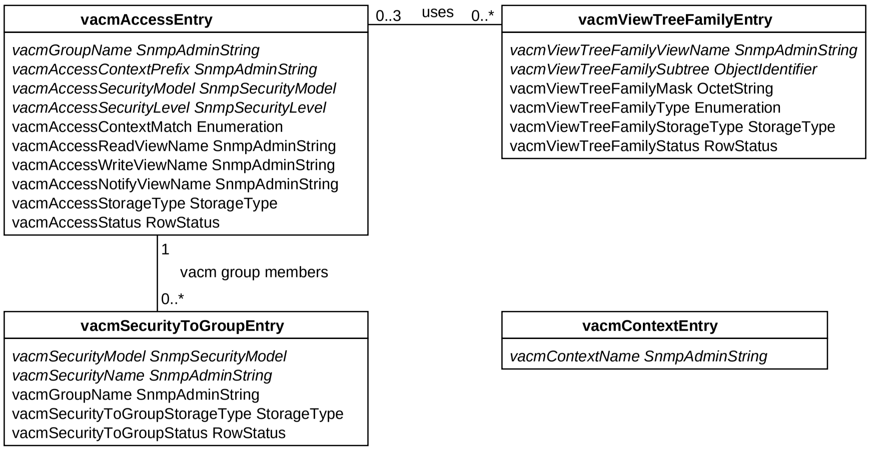
- Three different `securityLevels`: `noAuthNoPriv`, `authNoPriv`, `authPriv`
- A `securityName` is a security model independent name for a principal.

View-based Access Control Views (RFC 2275)



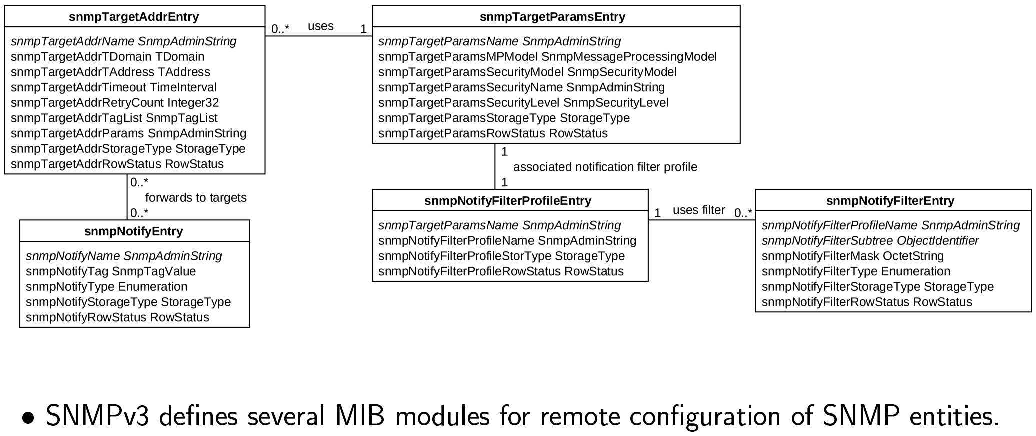
- A *view subtree* is a set of managed object instances with a common OID prefix.
- A *view tree family* is the combination of an OID prefix with a bit mask.
- A bit of the bit mask defines whether an OID prefix component is significant or not (wild-carding).
- A *view* is an ordered set of view tree families.
- Access control rights are defined by a *read view*, *write view* or *notify view*.

View-based Access Control MIB (RFC 2275)



- A security name (with a given security level) can not be a member of multiple groups.
- The `vacmViewTreeFamilyType` can be used to include or exclude a view tree family.
- The context table is kind of degenerated.

Remote Configuration of SNMP Entities



- SNMPv3 defines several MIB modules for remote configuration of SNMP entities.
- Several of these MIBs can be used with SNMPv1 through coexistence mappings.
- The example shows the structure of MIB tables for notification forwarding and notification filtering.

Coexistence with SNMPv1 (and SNMPv2c)

- Community-based message processing model:
  - Integration of SNMPv1 (SNMPv2c) into the SNMP architecture.
  - Definition of a MIB for remote configuration of SNMPv1 (SNMPv2c) agents.
- Error code and exception mappings:
  - Mappings of SNMPv3 error codes and exceptions into SNMPv1 error codes.
- Handling of unsupported data types:
  - Unsupported data types are implicitly not in view.
- Conversion of trap messages:
  - All information contained in a SNMPv1 trap can be mapped into a SNMPv3 trap.
- SMI conversion from SMIv1 to SMIv2:
  - Guidelines for converting SMIv1 MIB modules into SMIv2 MIB modules.

SNMP Community MIB

snmpCommunityEntry
snmpCommunityIndex SnmpEngineID
snmpCommunityName OctetString
snmpCommunitySecurityName SnmpAdminString
snmpCommunityContextEngineID SnmpEngineID
snmpCommunityContextName SnmpAdminString
snmpCommunityTransportTag SnmpTagValue
snmpCommunityStorageType StorageType
snmpCommunityStatus RowStatus

snmpTargetAddrExtTable
snmpTargetAddrName SnmpAdminString
snmpTargetAddrTMask OctetString
snmpTargetAddrMMS Integer32

- Maps an SNMPv1 (SNMPv2c) community string to a `securityName` and a context (`contextEngineID` and a `contextName`).
- Messages may be filtered by the transport address.
- Provides an extension for the `snmpTargetAddrTable` with an address mask and a maximum message size.

Implementations, Products, Experiences

- Several implementations and products are available:

ACE*COMM	IBM
SNMP++v3 Project	InterWorking Labs
BMC Software	MG-SOFT Corporation
Cisco Systems	MultiPort Corporation
ISI/Epilogue	SNMP Research
Gambit Communications	TU Braunschweig
Halcyon	UC Davis
IBM Research	University of Quebec

  – Visit the SNMPv3 Web page for more information.
  (`http://www.ibr.cs.tu-bs.de/projects/snmpv3/`)

- Experiences:
  – Configuring VACM manually is an error prone and time consuming task.
  – Remote configuration and key management requires not trivial applications.

Known Problems and Limitations of SNMPv3

- Missing extensibility for new base data types (e.g. Unsigned64).
- Missing extensibility for new protocol operations (e.g. GetSubtree).
- Limited flexibility for the definition of VACM rules.
- Asymmetries between notification filtering and VACM filtering.
- Positioning of security information in the middle of the message.
- Strength of USM security (DES versus Tripple-DES, key change procedure).
- Unnecessary complexity and misleading names in the message format definition.
- Insufficient performance gains compared to SNMPv1 (bulk data transfer).
- Degrees of freedom in complex write operations on tables are likely to cause interoperability problems.

2. Agent Extensibility Protocol (AgentX)

2.1 Motivation and Historic Perspective

2.2 Protocol Operations

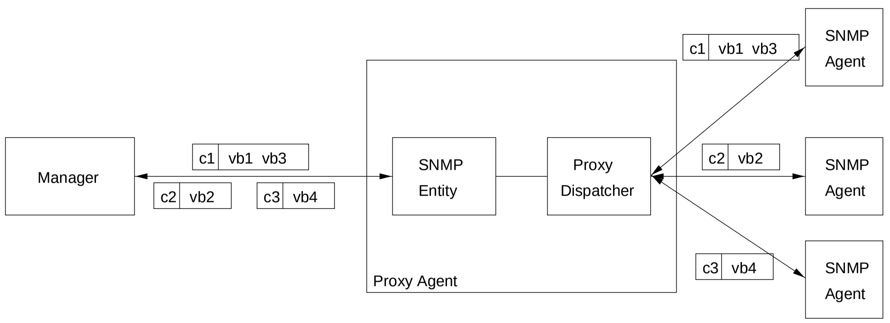
2.3 Transport Mappings and Encodings

2.4 Registrations

2.5 Index Allocation

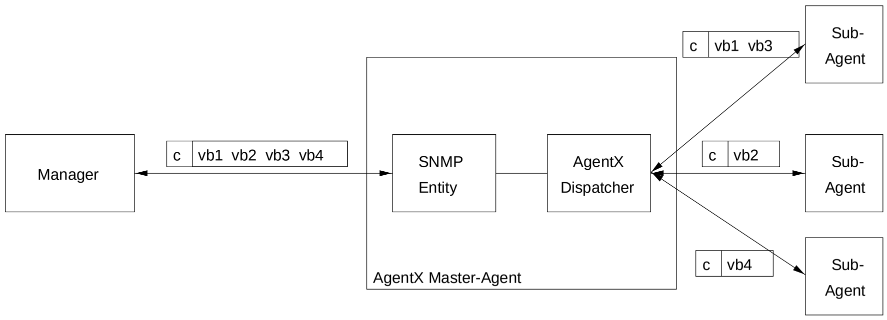
2.6 Implementations, Products, Experiences

Proxy Agents



- Proxies provide access to SNMP engines that are not directly accessible (e.g. Firewalls).
- Management applications must address the target agent through the proxy agent (no transparency).
- Proxies can translate between different protocol versions.
- Proxies are fully specified in the SNMPv3 specifications.
- Limited efficiency since all parties must implement/use a full SNMP protocol entity.

Extensible Agents



- Separation of the SNMP protocol engine from the MIB instrumentation.
- MIB module implementations can be added dynamically.
- Management applications have transparent access to "distributed agents."
- An extensibility protocol manages communication between master and sub-agents.
- Requires relatively complex operations on the master agent side in order to realize SNMP lexicographic ordering and access control efficiently.

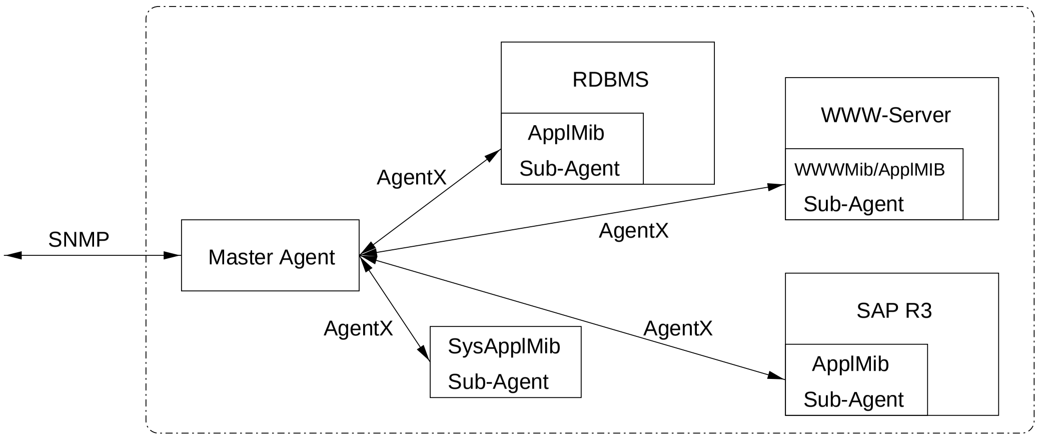
History of Extensible Agent Technology

- SNMP MUX Protocol (SMUX) circa 1991 (RFC 1227)
- SNMP Distributed Protocol Interface (DPI) circa 1991–1994 (RFC 1592)
- Digital's extensible SNMP agent (eSNMP) circa 1995
- SNMP Research's Emanate
- Several other proprietary protocols, often derived from SMUX
- Agent Extensibility Protocol (AgentX) circa 1998 (RFC 2257)

Agent Extensibility Protocol (AgentX) Version 1 (RFC 2257)

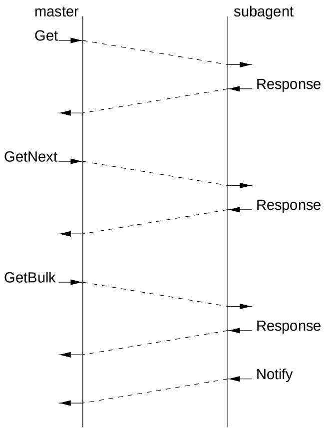
- Open standard for the implementation of extensible sub-agents.
- Based on experiences with several similar non-standard protocols.
- Core technology for modular networking devices.
- Required for portable system and application management agents.
- The AgentX master agent is MIB ignorant and SNMP omniscient.
- The AgentX sub-agent is SNMP ignorant and MIB omniscient.
- AgentX supports sub-agent integration through index allocation.
- Efficient AgentX message formats and encodings.

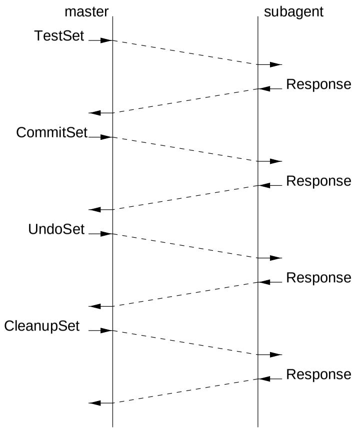
Example: Application Management with AgentX



- Application management MIBs require instrumentation in the applications.
- AgentX provides the infrastructure for implementing application management MIBs.

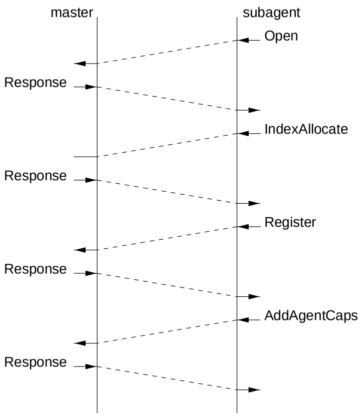
AgentX Protocol Operations (RFC 2257)

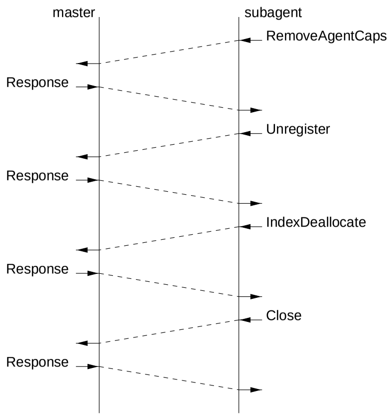




- SNMP Get, GetNext, GetBulk PDUs are mapped to similar AgentX PDUs.
- AgentX employs test-commit-undo-cleanup phases to achieve "as if simultaneous" semantics for SNMP write operations.
- The AgentX Notify operation may be mapped to SNMP Trap or Inform messages.

Administrative AgentX Protocol Operations (RFC 2257)





- Administrative protocol operations are used to open and close AgentX sessions, allocate and deallocate index values, register and unregister MIB regions exported by sub-agents, and add and remove agent capabilities (`sysOrTable`).

Transport Mappings and Encodings

- AgentX defines the following transport mappings:
  - AgentX over TCP (well known port number 705)
  - UNIX domain sockets (well known socket endpoint /var/agentx/master)
- Additional transport mappings (e.g. via shared memory) can be added.
- AgentX does not use BER encodings for efficiency reasons.
- AgentX encodings provide natural alignment with respect to the start of a PDU.
- AgentX sub-agents may use the native byte ordering.
- AgentX encodes well-known object identifier prefixes efficiently.
- OCTET STRING value are padded to align the encoded value to 4 byte boundaries.
- AgentX introduces search ranges which consist of two OID values.

OID Registration and Scoping

- Sub-agents may register single instances:
  (e.g. 1.3.6.1.2.1.25.1.2.0 = HOST-RESOURCES-MIB.hrSystemDate.0)
- Sub-agents may register OID regions:
  (e.g. 1.3.6.1.2.1.2.2.1.[1-22].7 = IF-MIB.ifIndex.7 – IF-MIB.ifSpecific.7)
- Only a single sub-agent can be "authoritative" for a particular OID region.
- Priority values are used to identify the authoritative sub-agent if regions overlap.
- Search ranges may be used during GetNext/GetBulk processing (scopes).
- Retrieving the next instance from an "authoritative" sub-agent may result in a loop over MIB subtrees.

Index Allocation

- A shared MIB table contains rows that are implemented by different sub-agents.
- Index values may be allocated to support sub-agents implementing shared tables.
- A sub-agent may request the allocation of:
  - a specific index value
  - an index value that is not currently allocated
  - an index value that has never been allocated
- The master agent maintains a database of index objects (OIDs), and, for each index, the values that have been allocated for it.
- Example:
  - A sub-agent wants to export a row in the IF-MIB.ifTable.
  - It requests an allocation of a value for ifIndex that has never been allocated.
  - The master agent returns the value 7.
  - It now attempts to register row 7 of the IF-MIB.ifTable (1.3.6.1.2.1.2.2.1.[1-22].7).

Implementations, Products, Experiences

- Implementations at various stages of completion:

ACE*COMM	ISI/Epilogue
BMC Software	MG-SOFT
Carnegie Mellon University	MultiPort Corporation
Compaq/DEC	SNMP Research

  - Visit the AgentX Web page for more information. (`http://www.scguild.com/agentx/`)
- Experiences:
  - Interactions with VACM access control can result in slow responses.
  - Some distributed management MIBs need to check access rights on other objects.
  - No support for index arbitration between sub-agents.
  - Not yet shipped by the average operating system vendor – but this may change!

3. Distributed Internet Management

3.1 Expression MIB

3.2 Event MIB

3.3 Scheduling MIB

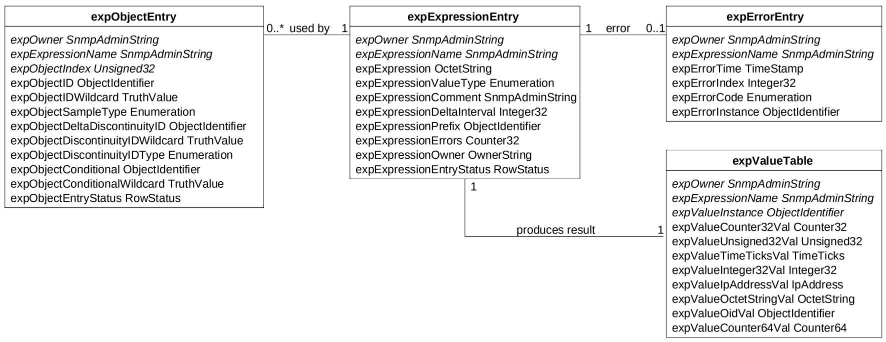
3.4 Script MIB

3.5 Remote Operations MIB

3.6 Notification Logging MIB

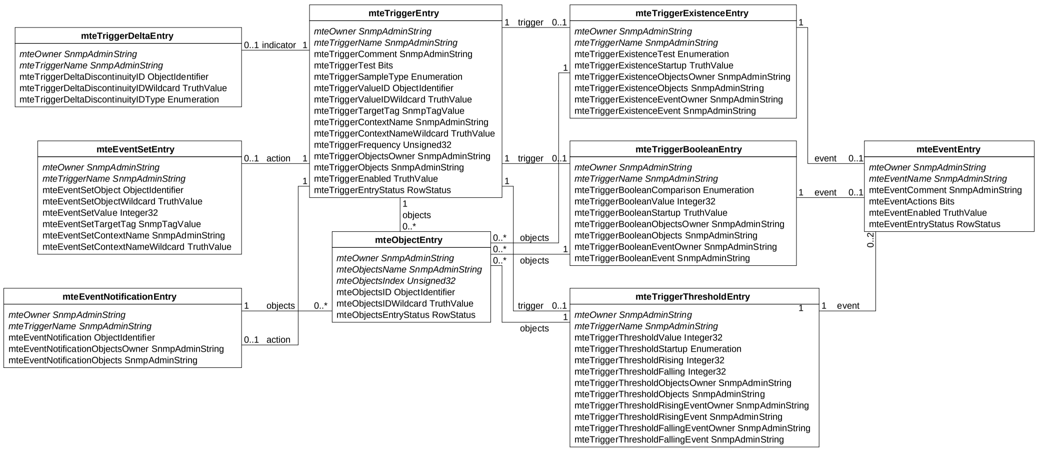
3.7 Implementations, Products, Experiences

Structure of the Expression MIB



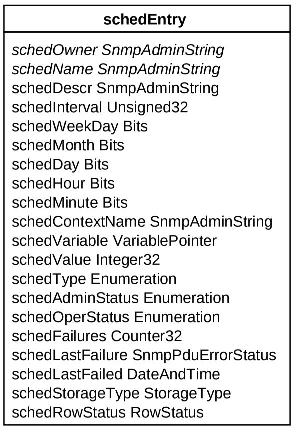
- Computation of expressions over MIB variables.
- Wildcarding can be used to apply a single expression to a complete table.
- Results can be used in other expressions as long as the expression is not recursive.
- Expressions are intended to operate on local MIB data.
- Expressions over counter objects require continuous sampling and state information.

Structure of the Event MIB



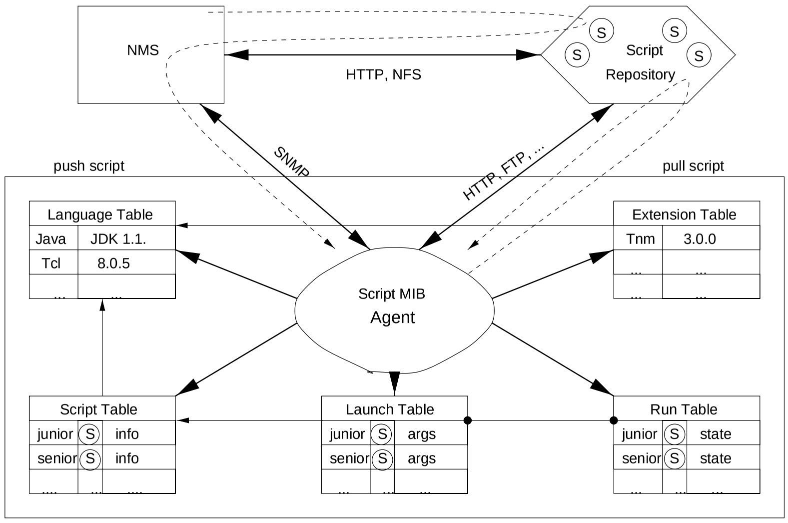
- Generation of an event if a MIB variable changes or crosses thresholds.
- Triggers define the conditions that lead to events.
- Events may cause actions such as notifications or set operations.
- Triggers using counter objects require continuous sampling and state information.

Structure of the Scheduling MIB

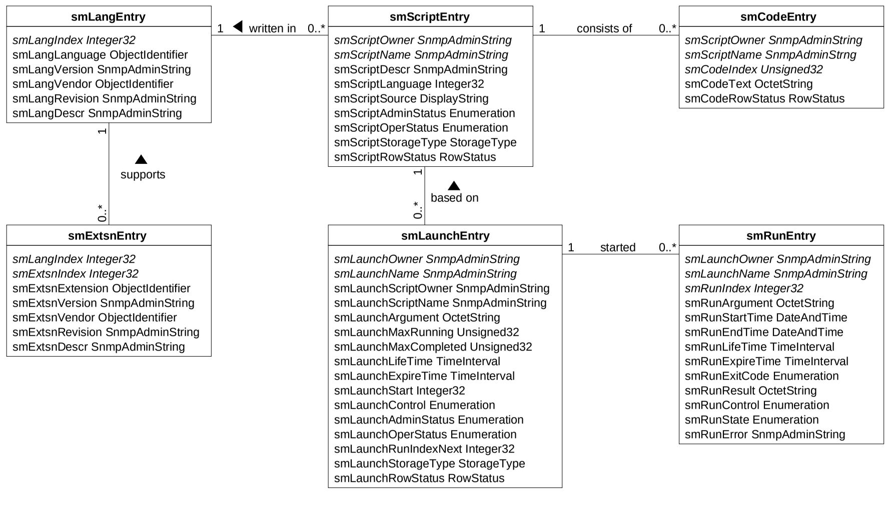


- Periodic or calendar driven set operations on MIB variables.
- One-shot schedules are calendar driven schedules that fire only once.
- Can be used to trigger management functions or to change the status of resources.
- The target MIB variable must be compatible with the type Integer32.

Management by Delegation with the Script MIB

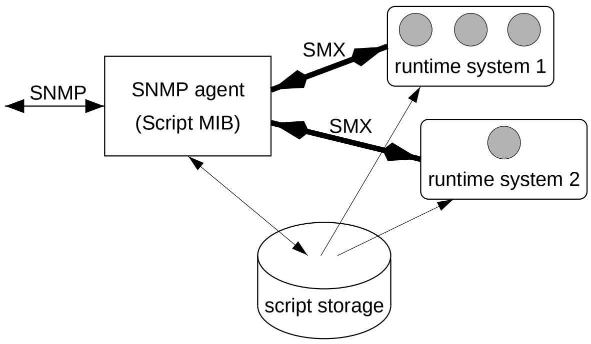


Structure of the Script MIB



- The `smCodeTable` is optional and not required for conformance.
- The `smLangTable` and `smExtsnTable` are read-only.

Script MIB Extensibility Protocol (SMX)

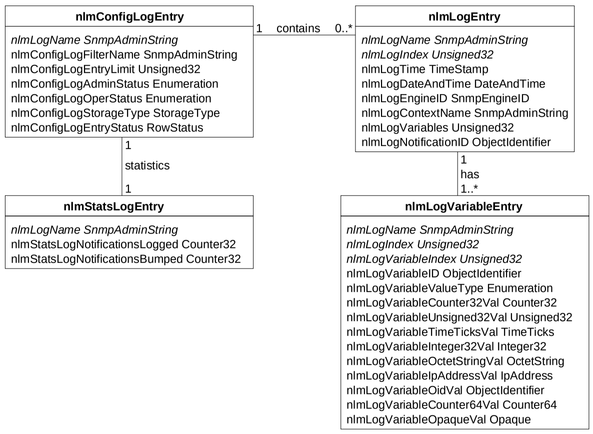


- Separates language specific runtime systems from the runtime system independent Script MIB implementation.
- Supports multiple languages and runtimes with different security profiles.
- Simple protocol running over a local TCP connection.
- Connection establishment procedure verifies a security cookie.
- Uses the local file system to pass executable code to runtime systems.

Remote Operations MIBs

- Remote ping MIB:
  - Enables a management application to perform a ping from a remote host.
- Remote traceroute MIB:
  - Enables a management application to perform a traceroute from a remote host.
- Remote name lookup MIB:
  - Enables a management application to perform a name lookup from a remote host.
- History tables are provided so that ping and traceroute tests can run offline.
- These MIBs are still under discussion.

Structure of the Notification Log MIB



- Common infrastructure in the form of a local logging function.
- It is intended primarily for senders of notifications.
- Applications can poll the log to know that they have not missed notifications.

Implementations, Products, Experiences

- One publically known implementation of the Expression MIB.
- CISCO plans to ship the Expression and Event MIB in products.
- Three independant publically known implementations of the Script MIB.
- Work towards real-world applications of the Script MIB.
- Interest in the Script MIB prototype by some European telecoms.
- Some enhancements of the Script MIB expected during the year 1999.

4. Summary and Perspective

4.1 Putting Everything Together

4.2 Next Generation Structure of Management Information (SMIng)

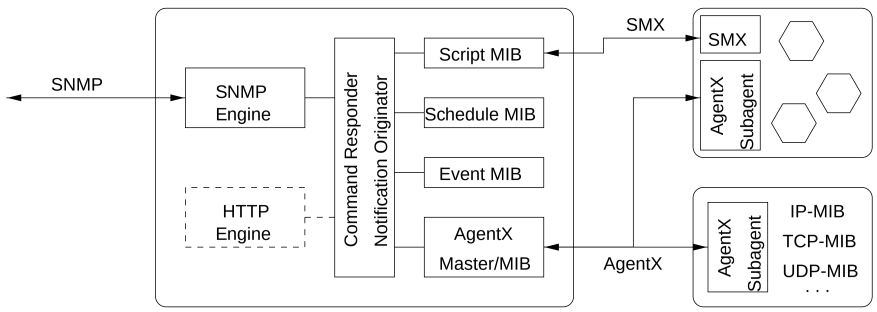
4.3 Bulk MIB Data Transfers

4.4 Future of Internet Management

4.5 References

4.6 Links to Online Resources

Secure, Modular and Extensible SNMP Agents



- SNMPv3 provides message security and authorization services.
- AgentX provides the interface and services for dynamic agent extensions.
- Distributed management MIBs realize the control infrastructure.

⇒ No documented "manager extensibility" protocol yet for allowing "MIBlets" to use the SNMP engine as a command generator or notification receiver.

Structure of Management Information Version 2 (SMIv2)

- Status of SMIv2:
  - Revised in 1998 by a "Design Team" (Clarifications, Corrections).
  - Publication as Internet Standard 58 (RFC 2578 – 2580) in April 1999.
  - SMIv2 will replace SMIv1 once all relevant IETF SMIv1 MIBS have been converted.
  - Good support from SNMP vendors and several (public) implementations.
  - Basis for MIB translators to other frameworks (e.g. CORBA, JDMK).

- Some remaining problems of SMIv2:
  - SMIv2 specification still depends on ASN.1 (1988).
  - Lack of a BNF definition makes it hard to build good parsers.
  - No support for new base types (e.g. Integer64, Unsigned64).
  - No extensibility of the SMIv2 language.
  - Several minor oddities (e.g. no UNITS clause in TEXTUAL-CONVENTIONS).
  - Unneeded complexity (e.g. arbitrary forward references, duplicate definitions).

Next Generation Structure of Management Information (SMIng)

- Design Goals:
  - SMIng should be independent from any other "external" standards.
  - SMIng specification written in the augmented BNF (ABNF).
  - SMIng statements follow a small set of core rules.
  - SMIng simplifies the implementation of parsers.
  - SMIng keeps related definitions close together to enhance readability.
  - SMIng has a complete set of base data types (64 bit and floating point numbers).
  - SMIng supports future extensions through the extension and unknown statement.

- Implementation of a "C" library to access MIB definitions (`libsmi`):
  - Several backend parsers for SMIv1, SMIv2 and SMIng.
  - Applications using `libsmi`: `smidump`, `smilint`, ...

⇒ Research project at the Technical University of Braunschweig.

Efficient Bulk Transfer of MIB-Data

- Approach #1: SNMP extensions for bulk MIB data transfers
  - Use TCP as a transport protocol.
  - Compression of SNMP messages using `gzip`.
  - Introduction of a new `GetSubtree` protocol operation.
- Approach #2: SNMP in conjunction with FTP
  - Definition of a MIB for storing MIB data in local files.
  - Definition of a MIB for initiating FTP transfers.
- Approach #3: Alternate protocols
  - Definition of a MIME type for carrying MIB data.
  - Transfer of MIME encapsulated MIB data via HTTP or SMTP.

⇒ The Network Management Research Group (NMRG) is working on approach #1.

Future of Internet Management

- Things that may be useful (short term):
  - Standardized APIs for SNMP and for accessing MIB definitions?
  - Protocols and APIs for exchanging topology and configuration information?
  - Protocols and APIs for exchanging alarm and trouble ticket records?
  - SNMP version 4 (really?)
  - Alternate protocols to exchange management information?
- Longer term perspective:
  - Less is more! $\Longrightarrow$ Self-managing devices and networks?
  - What are the alternatives? CORBA? CIM? CMIP/GDMO/TMN?
  - What about active networks and intelligent mobile agents?

Request for Comments (RFCs)

- D. Harrington, R. Presuhn, B. Wijnen: An Architecture for Describing SNMP Management Frameworks, RFC 2271, January 1998
- J. Case, D. Harrington, R. Presuhn, and B. Wijnen, Message Processing and Dispatching for the Simple Network Management Protocol (SNMP), RFC 2272, January 1998
- D. Levi, P. Meyer, B. Stewart: SNMPv3 Applications, RFC 2273, January 1998
- U. Blumenthal, B. Wijnen: User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), RFC 2274, January 1998.
- B. Wijnen, R. Presuhn, K. McCloghrie: View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP), RFC 2275, January 1998.
- J. Case, K. McCloghrie, M. Rose, S. Waldbusser, Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1905, January 1996
- J. Case, K. McCloghrie, M. Rose, S. Waldbusser, Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1906, January 1996
- J. Case, K. McCloghrie, M. Rose, S. Waldbusser, Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1907, January 1996
- J. Case, R. Mundy, D. Partain, B. Stewart: Introduction to Version 3 of the Internet-standard Network Management Framework, RFC (to be published), 1999
- R. Frye, D. Levi, S. Routhier, B. Wijnen: Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework, RFC (to be published), 1999

Request for Comments (RFCs)

- K. McCloghrie, D. Perkins, J. Schönwälder, J. Case, M. Rose, S. Waldbusser: Structure of Management Information Version 2 (SMIv2), STD 58, RFC 2578, April 1999
- K. McCloghrie, D. Perkins, J. Schönwälder, J. Case, M. Rose, S. Waldbusser: Textual Conventions for SMIv2, STD 58, RFC 2579, April 1999
- K. McCloghrie, D. Perkins, J. Schönwälder, J. Case, M. Rose, S. Waldbusser: Conformance Statements for SMIv2, STD 58, RFC 2580, April 1999
- M. Daniele, B. Wijnen, and D. Francisco: Agent Extensibility (AgentX) Protocol Version 1, RFC 2257, January 1998
- D.B. Levi and J. Schönwälder: Definitions of Managed Objects for the Delegation of Management Scripts, RFC (to be published), 1999
- D.B. Levi and J. Schönwälder: Definitions of Managed Objects for Scheduling Management Operations, RFC (to be published), 1999
- J. Schönwälder, J. Quittek: Script MIB Extensibility Protocol Version 1.0, RFC (to be published), 1999
- B. Stewart, Expression MIB, (work in progress), 1999
- B. Stewart, Event MIB, (work in progress), 1999
- B. Stewart, Notification Log MIB, (work in progress), 1999
- K. White, Definitions of Managed Objects for Remote Ping, Traceroute, and Name Lookup Operations, (work in progress), 1999

Books and Articles

- W. Stallings: SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, Addison-Wesley, 1999
- D. Zeltserman: A Practical Guide to SNMPv3 and Network Management, Prentice Hall, 1999
- D. Perkins and E. McGinnis: Understanding SNMP MIBs, Prentice Hall, 1997
- The SimpleTimes, Special Issue on Agent Extensibility, SimpleTimes 4(2), April 1996
- The SimpleTimes, Special Issue on SNMP Version 3, SimpleTimes 5(1), December 1997
- M. White, S. Gudur: An Overview of the AgentX Protocol, SimpleTimes 6(1), April 1998
- U. Blumenthal, N.C. Hien, B. Wijnen: Key derivation for network management applications, IEEE Network Magazine, 11(4), 1997
- J. Schönwälder: Network Management by Delegation - From Research Prototypes Towards Standards, Computer Networks and ISDN Systems, 29(15), November 1997
- J. Schönwälder, J. Quittek: Secure Management By Delegation within the Internet Management Framework, 6th International Symposium on Integrated Network Management, Boston, May 1999

Links to Online Information

- Internet Engineering Task Force (IETF) `http://www.ietf.org/`
- Internet Research Task Force (IETF) `http://www.irtf.org/`
- The SimpleTimes `http://www.simple-times.org/`
- The Simple Web `http://wwwsnmp.cs.utwente.nl/`
- IETF OPS Area `mailto:ops-area@ops.ietf.org`
- Tutorial Slides `http://www.ibr.cs.tu-bs.de/~schoenw/`