

Administrative Model
for version 2 of the
Simple Network Management Protocol (SNMPv2)

Status of this Memo

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

Table of Contents

| | |
|--|----|
| 1 Introduction | 2 |
| 1.1 A Note on Terminology | 2 |
| 2 Elements of the Model | 3 |
| 2.1 SNMPv2 Party | 3 |
| 2.2 SNMPv2 Entity | 6 |
| 2.3 SNMPv2 Management Station | 7 |
| 2.4 SNMPv2 Agent | 7 |
| 2.5 View Subtree | 7 |
| 2.6 MIB View | 8 |
| 2.7 Proxy Relationship | 8 |
| 2.8 SNMPv2 Context | 10 |
| 2.9 SNMPv2 Management Communication | 10 |
| 2.10 SNMPv2 Authenticated Management Communication | 12 |
| 2.11 SNMPv2 Private Management Communication | 13 |
| 2.12 SNMPv2 Management Communication Class | 14 |
| 2.13 SNMPv2 Access Control Policy | 14 |
| 3 Elements of Procedure | 17 |
| 3.1 Generating a Request | 17 |
| 3.2 Processing a Received Communication | 18 |
| 3.3 Generating a Response | 21 |

| | |
|--|----|
| 4 Application of the Model | 23 |
| 4.1 Non-Secure Minimal Agent Configuration | 23 |
| 4.2 Secure Minimal Agent Configuration | 26 |
| 4.3 MIB View Configurations | 28 |
| 4.4 Proxy Configuration | 32 |
| 4.4.1 Foreign Proxy Configuration | 33 |
| 4.4.2 Native Proxy Configuration | 37 |
| 4.5 Public Key Configuration | 41 |
| 5 Security Considerations | 44 |
| 6 Acknowledgements | 45 |
| 7 References | 46 |
| 8 Authors' Addresses | 47 |

1. Introduction

A network management system contains: several (potentially many) nodes, each with a processing entity, termed an agent, which has access to management instrumentation; at least one management station; and, a management protocol, used to convey management information between the agents and management stations. Operations of the protocol are carried out under an administrative framework which defines both authentication and authorization policies.

Network management stations execute management applications which monitor and control network elements. Network elements are devices such as hosts, routers, terminal servers, etc., which are monitored and controlled through access to their management information.

It is the purpose of this document, the Administrative Model for SNMPv2, to define how the administrative framework is applied to realize effective network management in a variety of configurations and environments.

The model described here entails the use of distinct identities for peers that exchange SNMPv2 messages. Thus, it represents a departure from the community-based administrative model of the original SNMP [1]. By unambiguously identifying the source and intended recipient of each SNMPv2 message, this new strategy improves upon the historical community scheme both by supporting a more convenient access control model and allowing for effective use of asymmetric (public key) security protocols in the future.

1.1. A Note on Terminology

For the purpose of exposition, the original Internet-standard Network Management Framework, as described in RFCs 1155, 1157, and 1212, is termed the SNMP version 1 framework (SNMPv1). The current framework is termed the SNMP version 2 framework (SNMPv2).

2. Elements of the Model

2.1. SNMPv2 Party

A SNMPv2 party is a conceptual, virtual execution environment whose operation is restricted (for security or other purposes) to an administratively defined subset of all possible operations of a particular SNMPv2 entity (see Section 2.2). Whenever a SNMPv2 entity processes a SNMPv2 message, it does so by acting as a SNMPv2 party and is thereby restricted to the set of operations defined for that party. The set of possible operations specified for a SNMPv2 party may be overlapping or disjoint with respect to the sets of other SNMPv2 parties; it may also be a proper or improper subset of all possible operations of the SNMPv2 entity.

Architecturally, each SNMPv2 party comprises

- o a single, unique party identity,
- o a logical network location at which the party executes, characterized by a transport protocol domain and transport addressing information,
- o a single authentication protocol and associated parameters by which all protocol messages originated by the party are authenticated as to origin and integrity, and
- o a single privacy protocol and associated parameters by which all protocol messages received by the party are protected from disclosure.

Conceptually, each SNMPv2 party may be represented by an ASN.1 value with the following syntax:

```
SnmpParty ::= SEQUENCE {  
    partyIdentity  
        OBJECT IDENTIFIER,  
    partyTDomain  
        OBJECT IDENTIFIER,  
    partyTAddress  
        OCTET STRING,  
    partyMaxMessageSize  
        INTEGER,  
    partyAuthProtocol  
        OBJECT IDENTIFIER,  
    partyAuthClock  
        INTEGER,  
    partyAuthPrivate  
        OCTET STRING,  
    partyAuthPublic  
        OCTET STRING,  
    partyAuthLifetime  
        INTEGER,  
    partyPrivProtocol  
        OBJECT IDENTIFIER,  
    partyPrivPrivate  
        OCTET STRING,  
    partyPrivPublic  
        OCTET STRING  
}
```

For each SnmpParty value that represents a SNMPv2 party, the following statements are true:

- o Its partyIdentity component is the party identity.
- o Its partyTDomain component is called the transport domain and indicates the kind of transport service by which the party receives network management traffic. An example of a transport domain is snmpUDPDDomain (SNMPv2 over UDP, using SNMPv2 parties).
- o Its partyTAddress component is called the transport addressing information and represents a transport service address by which the party receives network management traffic.

- o Its partyMaxMessageSize component is called the maximum message size and represents the length in octets of the largest SNMPv2 message this party is prepared to accept.
- o Its partyAuthProtocol component is called the authentication protocol and identifies a protocol and a mechanism by which all messages generated by the party are authenticated as to integrity and origin. In this context, the value noAuth signifies that messages generated by the party are not authenticated as to integrity and origin.
- o Its partyAuthClock component is called the authentication clock and represents a notion of the current time that is specific to the party. The significance of this component is specific to the authentication protocol.
- o Its partyAuthPrivate component is called the private authentication key and represents any secret value needed to support the authentication protocol. The significance of this component is specific to the authentication protocol.
- o Its partyAuthPublic component is called the public authentication key and represents any public value that may be needed to support the authentication protocol. The significance of this component is specific to the authentication protocol.
- o Its partyAuthLifetime component is called the lifetime and represents an administrative upper bound on acceptable delivery delay for protocol messages generated by the party. The significance of this component is specific to the authentication protocol.
- o Its partyPrivProtocol component is called the privacy protocol and identifies a protocol and a mechanism by which all protocol messages received by the party are protected from disclosure. In this context, the value noPriv signifies that messages received by the party are not protected from disclosure.
- o Its partyPrivPrivate component is called the private privacy key and represents any secret value needed to support the privacy protocol. The significance of this

component is specific to the privacy protocol.

- o Its partyPrivPublic component is called the public privacy key and represents any public value that may be needed to support the privacy protocol. The significance of this component is specific to the privacy protocol.

If, for all SNMPv2 parties realized by a SNMPv2 entity, the authentication protocol is noAuth and the privacy protocol is noPriv, then that entity is called non-secure.

2.2. SNMPv2 Entity

A SNMPv2 entity is an actual process which performs network management operations by generating and/or responding to SNMPv2 protocol messages in the manner specified in [2]. When a SNMPv2 entity is acting as a particular SNMPv2 party (see Section 2.1), the operation of that entity must be restricted to the subset of all possible operations that is administratively defined for that party.

By definition, the operation of a SNMPv2 entity requires no concurrency between processing of any single protocol message (by a particular SNMPv2 party) and processing of any other protocol message (by a potentially different SNMPv2 party). Accordingly, implementation of a SNMPv2 entity to support more than one party need not be multi-threaded. However, there may be situations where implementors may choose to use multi-threading.

Architecturally, every SNMPv2 entity maintains a local database that represents all SNMPv2 parties known to it - those whose operation is realized locally, those whose operation is realized by proxy interactions with remote parties or devices, and those whose operation is realized by remote entities. In addition, every SNMPv2 entity maintains a local database that represents all managed object resources (see Section 2.8) which are known to the SNMPv2 entity. Finally, every SNMPv2 entity maintains a local database that represents an access control policy (see Section 2.11) that defines the access privileges accorded to known SNMPv2 parties.

2.3. SNMPv2 Management Station

A SNMPv2 management station is the operational role assumed by a SNMPv2 party when it initiates SNMPv2 management operations by the generation of appropriate SNMPv2 protocol messages or when it receives and processes trap notifications.

Sometimes, the term SNMPv2 management station is applied to partial implementations of the SNMPv2 (in graphics workstations, for example) that focus upon this operational role. Such partial implementations may provide for convenient, local invocation of management services, but they may provide little or no support for performing SNMPv2 management operations on behalf of remote protocol users.

2.4. SNMPv2 Agent

A SNMPv2 agent is the operational role assumed by a SNMPv2 party when it performs SNMPv2 management operations in response to received SNMPv2 protocol messages such as those generated by a SNMPv2 management station (see Section 2.3).

Sometimes, the term SNMPv2 agent is applied to partial implementations of the SNMPv2 (in embedded systems, for example) that focus upon this operational role. Such partial implementations provide for realization of SNMPv2 management operations on behalf of remote users of management services, but they may provide little or no support for local invocation of such services.

2.5. View Subtree

A view subtree is the set of all MIB object instances which have a common ASN.1 OBJECT IDENTIFIER prefix to their names. A view subtree is identified by the OBJECT IDENTIFIER value which is the longest OBJECT IDENTIFIER prefix common to all (potential) MIB object instances in that subtree.

When the OBJECT IDENTIFIER prefix identifying a view subtree is longer than the OBJECT IDENTIFIER of an object type defined according to the SMI [3], then the use of such a view subtree for access control has granularity at the object instance level. Such granularity is considered beyond the scope of a

SNMPv2 entity acting in an agent role. As such, no implementation of a SNMPv2 entity acting in an agent role is required to support values of viewSubtree [6] which have more sub-identifiers than is necessary to identify a particular leaf object type. However, access control information is also used in determining which SNMPv2 entities acting in a manager role should receive trap notifications (Section 4.2.6 of [2]). As such, agent implementors might wish to provide instance-level granularity in order to allow a management station to use fine-grain configuration of trap notifications.

2.6. MIB View

A MIB view is a subset of the set of all instances of all object types defined according to the SMI [3] (i.e., of the universal set of all instances of all MIB objects), subject to the following constraints:

- o Each element of a MIB view is uniquely named by an ASN.1 OBJECT IDENTIFIER value. As such, identically named instances of a particular object type (e.g., in different agents) must be contained within different MIB views. That is, a particular object instance name resolves within a particular MIB view to at most one object instance.
- o Every MIB view is defined as a collection of view subtrees.

2.7. Proxy Relationship

A proxy relationship exists when, in order to process a received management request, a SNMPv2 entity must communicate with another, logically remote, entity. A SNMPv2 entity which processes management requests using a proxy relationship is termed a SNMPv2 proxy agent.

When communication between a logically remote party and a SNMPv2 entity is via the SNMPv2 (over any transport protocol), then the proxy party is called a SNMPv2 native proxy relationship. Deployment of SNMPv2 native proxy relationships is a means whereby the processing or bandwidth costs of management may be amortized or shifted - thereby facilitating

the construction of large management systems.

When communication between a logically remote party and a SNMPv2 entity party is not via the SNMPv2, then the proxy party is called a SNMPv2 foreign proxy relationship. Deployment of foreign proxy relationships is a means whereby otherwise unmanageable devices or portions of an internet may be managed via the SNMPv2.

The transparency principle that defines the behavior of a SNMPv2 entity in general applies in particular to a SNMPv2 proxy relationship:

The manner in which one SNMPv2 entity processes SNMPv2 protocol messages received from another SNMPv2 entity is entirely transparent to the latter.

The transparency principle derives directly from the historical SNMP philosophy of divorcing architecture from implementation. To this dichotomy are attributable many of the most valuable benefits in both the information and distribution models of the Internet-standard Network Management Framework, and it is the architectural cornerstone upon which large management systems may be built. Consistent with this philosophy, although the implementation of SNMPv2 proxy agents in certain environments may resemble that of a transport-layer bridge, this particular implementation strategy (or any other!) does not merit special recognition either in the SNMPv2 management architecture or in standard mechanisms for proxy administration.

Implicit in the transparency principle is the requirement that the semantics of SNMPv2 management operations are preserved between any two SNMPv2 peers. In particular, the "as if simultaneous" semantics of a Set operation are extremely difficult to guarantee if its scope extends to management information resident at multiple network locations. For this reason, proxy configurations that admit Set operations that apply to information at multiple locations are discouraged, although such operations are not explicitly precluded by the architecture in those rare cases where they might be supported in a conformant way.

Also implicit in the transparency principle is the requirement that, throughout its interaction with a proxy agent, a

management station is supplied with no information about the nature or progress of the proxy mechanisms by which its requests are realized. That is, it should seem to the management station - except for any distinction in underlying transport address - as if it were interacting via SNMPv2 directly with the proxied device. Thus, a timeout in the communication between a proxy agent and its proxied device should be represented as a timeout in the communication between the management station and the proxy agent. Similarly, an error response from a proxied device should - as much as possible - be represented by the corresponding error response in the interaction between the proxy agent and management station.

2.8. SNMPv2 Context

A SNMPv2 context is a collection of managed object resources accessible by a SNMPv2 entity. The object resources identified by a context are either local or remote.

A SNMPv2 context referring to local object resources is identified as a MIB view. In this case, a SNMPv2 entity uses local mechanisms to access the management information identified by the SNMPv2 context.

A remote SNMPv2 context referring to remote object resources is identified as a proxy relationship. In this case, a SNMPv2 entity acts as a proxy agent to access the management information identified by the SNMPv2 context.

2.9. SNMPv2 Management Communication

A SNMPv2 management communication is a communication from one specified SNMPv2 party to a second specified SNMPv2 party about management information that is contained in a SNMPv2 context accessible by the appropriate SNMPv2 entity. In particular, a SNMPv2 management communication may be

- o a query by the originating party about information accessible to the addressed party (e.g., getRequest, getNextRequest, or getBulkRequest),

- o an indicative assertion to the addressed party about information accessible to the originating party (e.g., Response, InformRequest, or SNMPv2-Trap),
- o an imperative assertion by the originating party about information accessible to the addressed party (e.g., setRequest), or
- o a confirmation to the addressed party about information received by the originating party (e.g., a Response confirming an InformRequest).

A management communication is represented by an ASN.1 value with the following syntax:

```
SnmpMgmtCom ::= [2] IMPLICIT SEQUENCE {  
    dstParty  
        OBJECT IDENTIFIER,  
    srcParty  
        OBJECT IDENTIFIER,  
    context  
        OBJECT IDENTIFIER,  
    pdu  
        PDUs  
}
```

For each SnmpMgmtCom value that represents a SNMPv2 management communication, the following statements are true:

- o Its dstParty component is called the destination and identifies the SNMPv2 party to which the communication is directed.
- o Its srcParty component is called the source and identifies the SNMPv2 party from which the communication is originated.
- o Its context component identifies the SNMPv2 context containing the management information referenced by the communication.
- o Its pdu component has the form and significance attributed to it in [2].

2.10. SNMPv2 Authenticated Management Communication

A SNMPv2 authenticated management communication is a SNMPv2 management communication (see Section 2.9) for which the originating SNMPv2 party is (possibly) reliably identified and for which the integrity of the transmission of the communication is (possibly) protected. An authenticated management communication is represented by an ASN.1 value with the following syntax:

```
SnmpAuthMsg ::= [1] IMPLICIT SEQUENCE {  
    authInfo  
        ANY, -- defined by authentication protocol  
    authData  
        SnmpMgmtCom  
}
```

For each SnmpAuthMsg value that represents a SNMPv2 authenticated management communication, the following statements are true:

- o Its authInfo component is called the authentication information and represents information required in support of the authentication protocol used by the SNMPv2 party originating the message. The detailed significance of the authentication information is specific to the authentication protocol in use; it has no effect on the application semantics of the communication other than its use by the authentication protocol in determining whether the communication is authentic or not.
- o Its authData component is called the authentication data and represents a SNMPv2 management communication.

2.11. SNMPv2 Private Management Communication

A SNMPv2 private management communication is a SNMPv2 authenticated management communication (see Section 2.10) that is (possibly) protected from disclosure. A private management communication is represented by an ASN.1 value with the following syntax:

```
SnmprPrivMsg ::= [1] IMPLICIT SEQUENCE {  
    privDst  
        OBJECT IDENTIFIER,  
    privData  
        [1] IMPLICIT OCTET STRING  
}
```

For each SmprPrivMsg value that represents a SNMPv2 private management communication, the following statements are true:

- o Its privDst component is called the privacy destination and identifies the SNMPv2 party to which the communication is directed.
- o Its privData component is called the privacy data and represents the (possibly encrypted) serialization (according to the conventions of [5]) of a SNMPv2 authenticated management communication (see Section 2.10).

2.12. SNMPv2 Management Communication Class

A SNMPv2 management communication class corresponds to a specific SNMPv2 PDU type defined in [2]. A management communication class is represented by an ASN.1 INTEGER value according to the type of the identifying PDU (see Table 1).

| | |
|-------------|-----|
| Get | 1 |
| GetNext | 2 |
| Response | 4 |
| Set | 8 |
| -- unused | 16 |
| GetBulk | 32 |
| Inform | 64 |
| SNMPv2-Trap | 128 |

Table 1: Management Communication Classes

The value by which a communication class is represented is computed as 2 raised to the value of the ASN.1 context-specific tag for the appropriate SNMPv2 PDU.

A set of management communication classes is represented by the ASN.1 INTEGER value that is the sum of the representations of the communication classes in that set. The null set is represented by the value zero.

2.13. SNMPv2 Access Control Policy

A SNMPv2 access control policy is a specification of a local access policy in terms of a SNMPv2 context and the management communication classes which are authorized between a pair of SNMPv2 parties. Architecturally, such a specification comprises four parts:

- o the targets of SNMPv2 access control - the SNMPv2 parties that may perform management operations as requested by management communications received from other parties,
- o the subjects of SNMPv2 access control - the SNMPv2 parties that may request, by sending management

communications to other parties, that management operations be performed,

- o the managed object resources of SNMPv2 access control - the SNMPv2 contexts which identify the management information on which requested management operations are to be performed, and
- o the policy that specifies the classes of SNMPv2 management communications pertaining to a particular SNMPv2 context that a particular target is authorized to accept from a particular subject.

Conceptually, a SNMPv2 access policy is represented by a collection of ASN.1 values with the following syntax:

```
AclEntry ::= SEQUENCE {  
    aclTarget  
        OBJECT IDENTIFIER,  
    aclSubject  
        OBJECT IDENTIFIER,  
    aclResources  
        OBJECT IDENTIFIER,  
    aclPrivileges  
        INTEGER  
}
```

For each such value that represents one part of a SNMPv2 access policy, the following statements are true:

- o Its `aclTarget` component is called the target and identifies the SNMPv2 party to which the partial policy permits access.
- o Its `aclSubject` component is called the subject and identifies the SNMPv2 party to which the partial policy grants privileges.
- o Its `aclResources` component is called the managed object resources and identifies the SNMPv2 context referenced by the partial policy.
- o Its `aclPrivileges` component is called the privileges and represents a set of SNMPv2 management communication classes which, when they reference the specified SNMPv2

context, are authorized to be processed by the specified target party when received from the specified subject party.

The application of SNMPv2 access control policy only occurs on receipt of management communications; it is not applied on transmission of management communications. Note, however, that ASN.1 values, having the syntax AclEntry, are also used in determining the destinations of a SNMPv2-Trap [2].

3. Elements of Procedure

This section describes the procedures followed by a SNMPv2 entity in processing SNMPv2 messages. These procedures are independent of the particular authentication and privacy protocols that may be in use.

3.1. Generating a Request

This section describes the procedure followed by a SNMPv2 entity whenever either a management request or a trap notification is to be transmitted by a SNMPv2 party.

- (1) A SnmpMgmtCom value is constructed for which the srcParty component identifies the originating party, for which the dstParty component identifies the receiving party, for which the context component identifies the desired SNMPv2 context, and for which the pdu component represents the desired management operation.
- (2) The local database of party information is consulted to determine the authentication protocol and other relevant information for the originating and receiving SNMPv2 parties.
- (3) A SnmpAuthMsg value is constructed with the following properties:

Its authInfo component is constructed according to the authentication protocol specified for the originating party.

In particular, if the authentication protocol for the originating SNMPv2 party is identified as noAuth, then this component corresponds to the OCTET STRING value of zero length.

Its authData component is the constructed SnmpMgmtCom value.

- (4) The local database of party information is consulted to determine the privacy protocol and other relevant information for the receiving SNMPv2 party.

- (5) A SnmpPrivMsg value is constructed with the following properties:

Its privDst component identifies the receiving SNMPv2 party.

Its privData component is the (possibly encrypted) serialization of the SnmpAuthMsg value according to the conventions of [5].

In particular, if the privacy protocol for the receiving SNMPv2 party is identified as noPriv, then the privData component is unencrypted. Otherwise, the privData component is processed according to the privacy protocol.

- (6) The constructed SnmpPrivMsg value is serialized according to the conventions of [5].
- (7) The serialized SnmpPrivMsg value is transmitted using the transport address and transport domain for the receiving SNMPv2 party.

Note that the above procedure does not include any application of any SNMPv2 access control policy (see section 2.13).

3.2. Processing a Received Communication

This section describes the procedure followed by a SNMPv2 entity whenever a management communication is received.

- (1) The snmpStatsPackets counter [7] is incremented. If the received message is not the serialization (according to the conventions of [5]) of an SnmpPrivMsg value, then that message is discarded without further processing. (If the first octet of the packet has the value hexadecimal 30, then the snmpStats30Something counter [7] is incremented prior to discarding the message; otherwise the snmpStatsEncodingErrors counter [7] is incremented.)
- (2) The local database of party information is consulted for information about the receiving SNMPv2 party identified by the privDst component of the SnmpPrivMsg value.

- (3) If information about the receiving SNMPv2 party is absent from the local database of party information, or indicates that the receiving party's operation is not realized by the local SNMPv2 entity, then the received message is discarded without further processing, after the snmpStatsUnknownDstParties counter [7] is incremented.
- (4) An ASN.1 OCTET STRING value is constructed (possibly by decryption, according to the privacy protocol in use) from the privData component of said SnmpPrivMsg value.

In particular, if the privacy protocol recorded for the party is noPriv, then the OCTET STRING value corresponds exactly to the privData component of the SnmpPrivMsg value.

- (5) If the OCTET STRING value is not the serialization (according to the conventions of [5]) of an SnmpAuthMsg value, then the received message is discarded without further processing, after the snmpStatsEncodingErrors counter [7] is incremented.
- (6) If the dstParty component of the authData component of the obtained SnmpAuthMsg value is not the same as the privDst component of the SnmpPrivMsg value, then the received message is discarded without further processing, after the snmpStatsDstPartyMismatches counter [7] is incremented.
- (7) The local database of party information is consulted for information about the originating SNMPv2 party identified by the srcParty component of the authData component of the SnmpAuthMsg value.
- (8) If information about the originating SNMPv2 party is absent from the local database of party information, then the received message is discarded without further processing, after the snmpStatsUnknownSrcParties counter [7] is incremented.
- (9) The obtained SnmpAuthMsg value is evaluated according to the authentication protocol and other relevant information associated with the originating and receiving SNMPv2 parties in the local database of party

information.

In particular, if the authentication protocol is identified as noAuth, then the SnmpAuthMsg value is always evaluated as authentic.

- (10) If the SnmpAuthMsg value is evaluated as unauthentic, then the received message is discarded without further processing, and if the snmpV2EnableAuthenTraps object [7] is enabled, then the SNMPv2 entity sends authorizationFailure traps [7] according to its configuration (Section 4.2.6 of [2]).
- (11) The SnmpMgmtCom value is extracted from the authData component of the SnmpAuthMsg value.
- (12) The local database of context information is consulted for information about the SNMPv2 context identified by the context component of the SnmpMgmtCom value.
- (13) If information about the SNMPv2 context is absent from the local database of context information, then the received message is discarded without further processing, after the snmpStatsUnknownContexts counter [7] is incremented.
- (14) The local database of access policy information is consulted for access privileges permitted by the local access policy to the originating SNMPv2 party with respect to the receiving SNMPv2 party and the indicated SNMPv2 context.
- (15) The management communication class is determined from the ASN.1 tag value associated with the PDUs component of the SnmpMgmtCom value. If the management information class of the received message is either 32, 8, 2, or 1 (i.e., GetBulk, Set, GetNext or Get) and the SNMPv2 context is not realized by the local SNMPv2 entity, then the received message is discarded without further processing, after the snmpStatsUnknownContexts counter [7] is incremented.
- (16) If the management communication class of the received message is either 128, 64 or 4 (i.e., SNMPv2-Trap, Inform, or Response) and this class is not among the

access privileges, then the received message is discarded without further processing, after the `snmpStatsBadOperations` counter [7] is incremented.

- (17) If the management communication class of the received message is not among the access privileges, then the received message is discarded without further processing after generation and transmission of a response message. This response message is directed to the originating SNMPv2 party on behalf of the receiving SNMPv2 party. Its context, var-bind-list and request-id components are identical to those of the received request. Its error-index component is zero and its error-status component is `authorizationError` [2].
- (18) If the SNMPv2 context refers to local object resources, then the management operation represented by the `SnmpMgmtCom` value is performed by the receiving SNMPv2 entity with respect to the MIB view identified by the SNMPv2 context according to the procedures set forth in [2].
- (19) If the SNMPv2 context refers to remote object resources, then the management operation represented by the `SnmpMgmtCom` value is performed through the appropriate proxy relationship.

3.3. Generating a Response

The procedure for generating a response to a SNMPv2 management request is identical to the procedure for transmitting a request (see Section 3.1), with these exceptions:

- (1) In Step 1, the `dstParty` component of the responding `SnmpMgmtCom` value is taken from the `srcParty` component of the original `SnmpMgmtCom` value; the `srcParty` component of the responding `SnmpMgmtCom` value is taken from the `dstParty` component of the original `SnmpMgmtCom` value; the context component of the responding `SnmpMgmtCom` value is taken from the context component of the original `SnmpMgmtCom` value; and, the pdu component of the responding `SnmpMgmtCom` value is the response which results from applying the operation specified in the original `SnmpMgmtCom` value.

- (2) In Step 7, the serialized SnmpPrivMsg value is transmitted using the transport address and transport domain from which its corresponding request originated - even if that is different from the transport information recorded in the local database of party information.

4. Application of the Model

This section describes how the administrative model set forth above is applied to realize effective network management in a variety of configurations and environments. Several types of administrative configurations are identified, and an example of each is presented.

4.1. Non-Secure Minimal Agent Configuration

This section presents an example configuration for a minimal, non-secure SNMPv2 agent that interacts with one or more SNMPv2 management stations. Table 2 presents information about SNMPv2 parties that is known both to the minimal agent and to the manager, while Table 3 presents similarly common information about the local access policy.

As represented in Table 2, the example agent party operates at UDP port 161 at IP address 1.2.3.4 using the party identity gracie; the example manager operates at UDP port 2001 at IP address 1.2.3.5 using the identity george. At minimum, a non-secure SNMPv2 agent implementation must provide for administrative configuration (and non-volatile storage) of the identities and transport addresses of two SNMPv2 parties: itself and a remote peer. Strictly speaking, other information about these two parties (including access policy information) need not be configurable.

| | | |
|---------------|-------------------|---------------------|
| Identity | gracie (agent) | george (manager) |
| Domain | snmpUDPDomain | snmpUDPDomain |
| Address | 1.2.3.4, 161 | 1.2.3.5, 2001 |
| Auth Prot | noAuth | noAuth |
| Auth Priv Key | " " | " " |
| Auth Pub Key | " " | " " |
| Auth Clock | 0 | 0 |
| Auth Lifetime | 0 | 0 |
| Priv Prot | noPriv | noPriv |
| Priv Priv Key | " " | " " |
| Priv Pub Key | " " | " " |

Table 2: Party Information for Minimal Agent

| Target | Subject | Context | Privileges |
|--------|---------|---------|------------------------------|
| gracie | george | local | 35 (Get, GetNext & GetBulk) |
| george | gracie | local | 132 (Response & SNMPv2-Trap) |

Table 3: Access Information for Minimal Agent

Suppose that the managing party george wishes to interrogate management information about the SNMPv2 context named "local" held by the agent named gracie by issuing a SNMPv2 GetNext request message. The manager consults its local database of party information. Because the authentication protocol for the party george is recorded as noAuth, the GetNext request message generated by the manager is not authenticated as to origin and integrity. Because, according to the manager's local database of party information, the privacy protocol for the party gracie is noPriv, the GetNext request message is not protected from disclosure. Rather, it is simply assembled, serialized, and transmitted to the transport address (IP address 1.2.3.4, UDP port 161) associated in the manager's local database of party information with the party gracie.

When the GetNext request message is received at the agent, the identity of the party to which it is directed (gracie) is

extracted from the message, and the receiving entity consults its local database of party information. Because the privacy protocol for the party gracie is recorded as noPriv, the received message is assumed not to be protected from disclosure. Similarly, the identity of the originating party (george) is extracted, and the local database of party information is consulted. Because the authentication protocol for the party george is recorded as noAuth, the received message is immediately accepted as authentic.

The received message is fully processed only if the agent's local database of access policy information authorizes GetNext request communications by the party george to the agent party gracie with respect to the SNMPv2 context "local". The database of access policy information presented as Table 3 authorizes such communications (as well as Get and GetBulk operations).

When the received request is processed, a Response message is generated which references the SNMPv2 context "local" and identifies gracie as the source party and george, the party from which the request originated, as the destination party. Because the authentication protocol for gracie is recorded in the local database of party information as noAuth, the generated Response message is not authenticated as to origin or integrity. Because, according to the local database of party information, the privacy protocol for the party george is noPriv, the response message is not protected from disclosure. The response message is transmitted to the transport address from which the corresponding request originated - without regard for the transport address associated with george in the local database of party information.

When the generated response is received by the manager, the identity of the party to which it is directed (george) is extracted from the message, and the manager consults its local database of party information. Because the privacy protocol for the party george is recorded as noPriv, the received response is assumed not to be protected from disclosure. Similarly, the identity of the originating party (gracie) is extracted, and the local database of party information is consulted. Because the authentication protocol for the party gracie is recorded as noAuth, the received response is immediately accepted as authentic.

The received message is fully processed only if the manager's local database of access policy information authorizes Response communications from the party *gracie* to the manager party *george* which reference the SNMPv2 context "local". The database of access policy information presented as Table 3 authorizes such Response messages (as well as SNMPv2-Trap messages).

4.2. Secure Minimal Agent Configuration

This section presents an example configuration for a secure, minimal SNMPv2 agent that interacts with a single SNMPv2 management station. Table 4 presents information about SNMPv2 parties that is known both to the minimal agent and to the manager, while Table 5 presents similarly common information about the local access policy.

The interaction of manager and agent in this configuration is very similar to that sketched above for the non-secure minimal agent - except that all protocol messages are authenticated as to origin and integrity and protected from disclosure. This example requires encryption in order to support distribution of secret keys via the SNMPv2 itself. A more elaborate example comprising an additional pair of SNMPv2 parties could support the exchange of non-secret information in authenticated messages without incurring the cost of encryption.

An actual secure agent configuration may require SNMPv2 parties for which the authentication and privacy protocols are *noAuth* and *noPriv*, respectively, in order to support clock synchronization (see [6]). For clarity, these additional parties are not represented in this example.

| | | |
|---------------|--------------------|--------------------|
| Identity | ollie | stan |
| | (agent) | (manager) |
| Domain | snmpUDPDomain | snmpUDPDomain |
| Address | 1.2.3.4, 161 | 1.2.3.5, 2001 |
| Auth Prot | v2md5AuthProtocol | v2md5AuthProtocol |
| Auth Priv Key | "0123456789ABCDEF" | "GHIJKL0123456789" |
| Auth Pub Key | " " | " " |
| Auth Clock | 0 | 0 |
| Auth Lifetime | 300 | 300 |
| Priv Prot | desPrivProtocol | desPrivProtocol |
| Priv Priv Key | "MNOPQR0123456789" | "STUVWX0123456789" |
| Priv Pub Key | " " | " " |

Table 4: Party Information for Secure Minimal Agent

| Target | Subject | Context | Privileges |
|--------|---------|---------|------------------------------|
| ollie | stan | local | 35 (Get, GetNext & GetBulk) |
| stan | ollie | local | 132 (Response & SNMPv2-Trap) |

Table 5: Access Information for Secure Minimal Agent

As represented in Table 4, the example agent party operates at UDP port 161 at IP address 1.2.3.4 using the party identity ollie; the example manager operates at UDP port 2001 at IP address 1.2.3.5 using the identity stan. At minimum, a secure SNMPv2 agent implementation must provide for administrative configuration (and non-volatile storage) of relevant information about two SNMPv2 parties: itself and a remote peer. Both ollie and stan authenticate all messages that they generate by using the SNMPv2 authentication protocol v2md5AuthProtocol and their distinct, private authentication keys. Although these private authentication key values ("0123456789ABCDEF" and "GHIJKL0123456789") are presented here for expository purposes, knowledge of private authentication keys is not normally afforded to human beings and is confined to those portions of the protocol implementation that require it.

When using the v2md5AuthProtocol, the public authentication key for each SNMPv2 party is never used in authentication and verification of SNMPv2 exchanges. Also, because the v2md5AuthProtocol is symmetric in character, the private authentication key for each party must be known to another SNMPv2 party with which authenticated communication is desired. In contrast, asymmetric (public key) authentication protocols would not depend upon sharing of a private key for their operation.

All protocol messages generated for transmission to the party stan are encrypted using the desPrivProtocol privacy protocol and the private key "STUVWX0123456789"; they are decrypted upon reception according to the same protocol and key. Similarly, all messages generated for transmission to the party ollie are encrypted using the desPrivProtocol protocol and private privacy key "MNOPQR0123456789"; they are correspondingly decrypted on reception. As with authentication keys, knowledge of private privacy keys is not normally afforded to human beings and is confined to those portions of the protocol implementation that require it.

4.3. MIB View Configurations

This section describes a convention for the definition of MIB views and, using that convention, presents example configurations of MIB views for SNMPv2 contexts that refer to local object resources.

A MIB view is defined by a collection of view subtrees (see Section 2.6), and any MIB view may be represented in this way. Because MIB view definitions may, in certain cases, comprise a very large number of view subtrees, a convention for abbreviating MIB view definitions is desirable.

The convention adopted in [4] supports abbreviation of MIB view definitions in terms of families of view subtrees that are either included in or excluded from the definition of the relevant MIB view. By this convention, a table locally maintained by each SNMPv2 entity defines the MIB view associated with each SNMPv2 context that refers to local object resources. Each entry in the table represents a family of view subtrees that (according to the type of that entry) is either included in or excluded from the MIB view of some

SNMPv2 context. Each table entry represents a subtree family as a pairing of an OBJECT IDENTIFIER value (called the family name) together with a bitstring value (called the family mask). The family mask indicates which sub-identifiers of the associated family name are significant to the definition of the represented subtree family. For each possible MIB object instance, that instance belongs to the view subtree family represented by a particular table entry if

- o the OBJECT IDENTIFIER name of that MIB object instance comprises at least as many sub-identifiers as does the family name for said table entry, and
- o each sub-identifier in the name of said MIB object instance matches the corresponding sub-identifier of the relevant family name whenever the corresponding bit of the associated family mask is non-zero.

The appearance of a MIB object instance in the MIB view for a particular SNMPv2 context is related to the membership of that instance in the subtree families associated with that SNMPv2 context in local table entries:

- o If a MIB object instance belongs to none of the relevant subtree families, then that instance is not in the MIB view for the relevant SNMPv2 context.
- o If a MIB object instance belongs to the subtree family represented by exactly one of the relevant table entries, then that instance is included in, or excluded from, the relevant MIB view according to the type of that entry.
- o If a MIB object instance belongs to the subtree families represented by more than one of the relevant table entries, then that instance is included in, or excluded from, the relevant MIB view according to the type of the single such table entry for which, first, the associated family name comprises the greatest number of sub-identifiers, and, second, the associated family name is lexicographically greatest.

The subtree family represented by a table entry for which the associated family mask is all ones corresponds to the single view subtree identified by the family name for that entry. Because the convention of [4] provides for implicit extension

of family mask values with ones, the subtree family represented by a table entry with a family mask of zero length always corresponds to a single view subtree.

| Context | Type | Family Name | Family Mask |
|---------|----------|-------------|-------------|
| lucy | included | internet | ''H |

Table 6: View Definition for Minimal Agent

Using this convention for abbreviating MIB view definitions, some of the most common definitions of MIB views may be conveniently expressed. For example, Table 6 illustrates the MIB view definitions required for a minimal SNMPv2 entity that having a single SNMPv2 context for which the associated MIB view embraces all instances of all MIB objects defined within the SNMPv2 Network Management Framework. The represented table has a single entry. The SNMPv2 context (lucy) for which that entry defines the MIB view is identified in the first column. The type of that entry (included) signifies that any MIB object instance belonging to the subtree family represented by that entry may appear in the MIB view for the SNMPv2 context lucy. The family name for that entry is internet, and the zero-length family mask value signifies that the relevant subtree family corresponds to the single view subtree rooted at that node.

Another example of MIB view definition (see Table 7) is that of a SNMPv2 entity having multiple SNMPv2 contexts with distinct MIB views. The MIB view associated with the SNMPv2 context lucy comprises all instances of all MIB objects defined within the SNMPv2 Network Management Framework, except those pertaining to the administration of SNMPv2 parties. In contrast, the MIB view attributed to the SNMPv2 context ricky contains only MIB object instances defined in the system group of the Internet-standard MIB together with those object instances by which SNMPv2 parties are administered.

| Context | Type | Family Name | Family Mask |
|---------|----------|-------------|-------------|
| lucy | included | internet | 'H |
| lucy | excluded | snmpParties | 'H |
| ricky | included | system | 'H |
| ricky | included | snmpParties | 'H |

Table 7: View Definition for Multiple Contexts

A more complicated example of MIB view configuration illustrates the abbreviation of related collections of view subtrees by view subtree families (see Table 8). In this example, the MIB view associated with the SNMPv2 context lucy includes all object instances in the system group of the Internet-standard MIB together with some information related to the second network interface attached to the managed device. However, this interface-related information does not include the speed of the interface. The family mask value 'FFA0'H in the second table entry signifies that a MIB object instance belongs to the relevant subtree family if the initial prefix of its name places it within the ifEntry portion of the registration hierarchy and if the eleventh sub-identifier of its name is 2. The MIB object instance representing the speed of the second network interface belongs to the subtree families represented by both the second and third entries of the table, but that particular instance is excluded from the MIB view for the SNMPv2 context lucy because the lexicographically greater of the relevant family names appears in the table entry with type excluded.

The MIB view for the SNMPv2 context ricky is also defined in this example. The MIB view attributed to the SNMPv2 context ricky includes all object instances in the icmp group of the Internet-standard MIB, together with all information relevant to the fifth network interface attached to the managed device. In addition, the MIB view attributed to the SNMPv2 context ricky includes the number of octets received on the fourth attached network interface.

| Context | Type | Family Name | Family Mask |
|---------|----------|------------------|-------------|
| lucy | included | system | 'H |
| lucy | included | { ifEntry 0 2 } | 'FFA0'H |
| lucy | excluded | { ifSpeed 2 } | 'H |
| ricky | included | icmp | 'H |
| ricky | included | { ifEntry 0 5 } | 'FFA0'H |
| ricky | included | { ifInOctets 4 } | 'H |

Table 8: More Elaborate View Definitions

While, as suggested by the examples above, a wide range of MIB view configurations are efficiently supported by the abbreviated representation of [4], prudent MIB design can sometimes further reduce the size and complexity of the most likely MIB view definitions. On one hand, it is critical that mechanisms for MIB view configuration impose no absolute constraints either upon the access policies of local administrations or upon the structure of MIB namespaces; on the other hand, where the most common access policies are known, the configuration costs of realizing those policies may be slightly reduced by assigning to distinct portions of the registration hierarchy those MIB objects for which local policies most frequently require distinct treatment.

4.4. Proxy Configuration

This section presents examples of SNMPv2 proxy configurations. On one hand, foreign proxy configurations provide the capability to manage non-SNMP devices. On the other hand, native proxy configurations allow an administrator to shift the computational burden of rich management functionality away from network devices whose primary task is not management. To the extent that SNMPv2 proxy agents function as points of aggregation for management information, proxy configurations may also reduce the bandwidth requirements of large-scale management activities.

The example configurations in this section are simplified for clarity: actual configurations may require additional parties in order to support clock synchronization and distribution of secrets.

4.4.1. Foreign Proxy Configuration

This section presents an example configuration by which a SNMPv2 management station may manage network elements that do not themselves support the SNMPv2. This configuration centers on a SNMPv2 proxy agent that realizes SNMPv2 management operations by interacting with a non-SNMPv2 device using a proprietary protocol.

Table 9 presents information about SNMPv2 parties that is recorded in the SNMPv2 proxy agent's local database of party information. Table 10 presents information about proxy relationships that is recorded in the SNMPv2 proxy agent's local database of context information. Table 11 presents information about SNMPv2 parties that is recorded in the SNMPv2 management station's local database of party information. Table 12 presents information about the database of access policy information specified by the local administration.

| | | | |
|---------------|----------------------|------------------------|----------------------|
| Identity | groucho (manager) | chico (proxy agent) | harpo (proxy dst) |
| Domain | snmpUDPDomain | snmpUDPDomain | acmeMgmtPrctl |
| Address | 1.2.3.4, 2002 | 1.2.3.5, 161 | 0x98765432 |
| Auth Prot | v2md5AuthProtocol | v2md5AuthProtocol | noAuth |
| Auth Priv Key | "0123456789ABCDEF" | "GHIJKL0123456789" | " " |
| Auth Pub Key | " " | " " | " " |
| Auth Clock | 0 | 0 | 0 |
| Auth Lifetime | 300 | 300 | 0 |
| Priv Prot | noPriv | noPriv | noPriv |
| Priv Priv Key | " " | " " | " " |
| Priv Pub Key | " " | " " | " " |

Table 9: Party Information for Proxy Agent

| Context | Proxy Destination | Proxy Source | Proxy Context |
|----------|-------------------|--------------|---------------|
| ducksoup | harpo | n/a | n/a |

Table 10: Proxy Relationships for Proxy Agent

| Identity | groucho (manager) | chico (proxy agent) |
|---------------|----------------------|------------------------|
| Domain | snmpUDPDomain | snmpUDPDomain |
| Address | 1.2.3.4, 2002 | 1.2.3.5, 161 |
| Auth Prot | v2md5AuthProtocol | v2md5AuthProtocol |
| Auth Priv Key | "0123456789ABCDEF" | "GHIJKL0123456789" |
| Auth Pub Key | " " | " " |
| Auth Clock | 0 | 0 |
| Auth Lifetime | 300 | 300 |
| Priv Prot | noPriv | noPriv |
| Priv Priv Key | " " | " " |
| Priv Pub Key | " " | " " |

Table 11: Party Information for Management Station

| Target | Subject | Context | Privileges |
|---------|---------|----------|------------------------------|
| chico | groucho | ducksoup | 35 (Get, GetNext & GetBulk) |
| groucho | chico | ducksoup | 132 (Response & SNMPv2-Trap) |

Table 12: Access Information for Foreign Proxy

As represented in Table 9, the proxy agent party operates at UDP port 161 at IP address 1.2.3.5 using the party identity chico; and, the example manager operates at UDP port 2002 at IP address 1.2.3.4 using the identity groucho. Both groucho and chico authenticate all messages that they generate by using the protocol v2md5AuthProtocol and their distinct, private authentication keys. Although these private authentication key values ("0123456789ABCDEF" and "GHIJKL0123456789") are presented here for expository

purposes, knowledge of private keys is not normally afforded to human beings and is confined to those portions of the protocol implementation that require it.

The party harpo does not send or receive SNMPv2 protocol messages; rather, all communication with that party proceeds via a hypothetical proprietary protocol identified by the value `acmeMgmtPrctl`. Because the party harpo does not participate in the SNMPv2, many of the attributes recorded for that party in the local database of party information are ignored.

Table 10 shows the proxy relationships known to the proxy agent. In particular, the SNMPv2 context `ducksoup` refers to a relationship that is satisfied by the party harpo. (The transport domain of the proxy destination party determines the interpretation of the proxy source and proxy context identities - in this case, use of the `acmeMgmtPrctl` indicates that the proxy source and context identities are ignored.)

In order to interrogate the proprietary device associated with the party harpo, the management station `groucho` constructs a SNMPv2 `GetNext` request contained within a `SnmpMgmtCom` value which references the SNMPv2 context `ducksoup`, and transmits it to the party chico operating (see Table 11) at UDP port 161, and IP address 1.2.3.5. This request is authenticated using the private authentication key "0123456789ABCDEF".

When that request is received by the party chico, the originator of the message is verified as being the party `groucho` by using local knowledge (see Table 9) of the private authentication key "0123456789ABCDEF". Because party `groucho` is authorized to issue `GetNext` (as well as `Get` and `GetBulk`) requests with respect to party chico and the SNMPv2 context `ducksoup` by the relevant access control policy (Table 12), the request is accepted. Because the local database of context information indicates that the SNMPv2 context `ducksoup` refers to a proxy relationship, the request is satisfied by its translation into appropriate operations of the `acmeMgmtPrctl` directed at party harpo. These new operations are transmitted to the party harpo at the address 0x98765432 in the `acmeMgmtPrctl` domain.

When and if the proprietary protocol exchange between the proxy agent and the proprietary device concludes, a SNMPv2

Response management operation is constructed by the SNMPv2 party chico to relay the results to party groucho again referring to the SNMPv2 context ducksoup. This response communication is authenticated as to origin and integrity using the authentication protocol v2md5AuthProtocol and private authentication key "GHIJKL0123456789" specified for transmissions from party chico. It is then transmitted to the SNMPv2 party groucho operating at the management station at IP address 1.2.3.4 and UDP port 2002 (the source address for the corresponding request).

When this response is received by the party groucho, the originator of the message is verified as being the party chico by using local knowledge (see Table 11) of the private authentication key "GHIJKL0123456789". Because party chico is authorized to issue Response communications with respect to party groucho and SNMPv2 context ducksoup by the relevant access control policy (Table 12), the response is accepted, and the interrogation of the proprietary device is complete.

It is especially useful to observe that the local database of party information recorded at the proxy agent (Table 9) need be neither static nor configured exclusively by the management station. For instance, suppose that, in this example, the acmeMgmtPrctl was a proprietary, MAC-layer mechanism for managing stations attached to a local area network. In such an environment, the SNMPv2 party chico would reside at a SNMPv2 proxy agent attached to such a LAN and could, by participating in the LAN protocols, detect the attachment and disconnection of various stations on the LAN. In this scenario, the SNMPv2 proxy agent could easily adjust its local database of party information to support indirect management of the LAN stations by the SNMPv2 management station. For each new LAN station detected, the SNMPv2 proxy agent would add to its local database of party information an entry analogous to that for party harpo (representing the new LAN station itself), and also add to its local database of context information an entry analogous to that for SNMPv2 context ducksoup (representing a proxy relationship for that new station in the SNMPv2 domain).

By using the SNMPv2 to interrogate the local database of party information held by the SNMPv2 proxy agent, a SNMPv2 management station can discover and interact with new stations as they are attached to the LAN.

4.4.2. Native Proxy Configuration

This section presents an example configuration that supports SNMPv2 native proxy operations - indirect interaction between a SNMPv2 agent and a management station that is mediated by a second SNMPv2 (proxy) agent.

This example configuration is similar to that presented in the discussion of SNMPv2 foreign proxy above. In this example, however, the party associated with the identity harpo receives messages via the SNMPv2, and, accordingly interacts with the SNMPv2 proxy agent chico using authenticated SNMPv2 communications.

Table 13 presents information about SNMPv2 parties that is recorded in the SNMPv2 proxy agent's local database of party information. Table 14 presents information about proxy relationships that is recorded in the SNMPv2 proxy agent's local database of context information. Table 11 presents information about SNMPv2 parties that is recorded in the SNMPv2 management station's local database of party information. Table 15 presents information about the database of access policy information specified by the local administration.

| | | |
|---------------|----------------------|------------------------|
| Identity | groucho (manager) | chico (proxy agent) |
| Domain | snmpUDPDomain | snmpUDPDomain |
| Address | 1.2.3.4, 2002 | 1.2.3.5, 161 |
| Auth Prot | v2md5AuthProtocol | v2md5AuthProtocol |
| Auth Priv Key | "0123456789ABCDEF" | "GHIJKL0123456789" |
| Auth Pub Key | " " | " " |
| Auth Clock | 0 | 0 |
| Auth Lifetime | 300 | 300 |
| Priv Prot | noPriv | noPriv |
| Priv Priv Key | " " | " " |
| Priv Pub Key | " " | " " |

| | | |
|---------------|----------------------|----------------------|
| Identity | harpo (proxy dst) | zeppo (proxy src) |
| Domain | snmpUDPDomain | snmpUDPDomain |
| Address | 1.2.3.6, 161 | 1.2.3.5, 161 |
| Auth Prot | v2md5AuthProtocol | v2md5AuthProtocol |
| Auth Priv Key | "MNOPQR0123456789" | "STUVWX0123456789" |
| Auth Pub Key | " " | " " |
| Auth Clock | 0 | 0 |
| Auth Lifetime | 300 | 300 |
| Priv Prot | noPriv | noPriv |
| Priv Priv Key | " " | " " |
| Priv Pub Key | " " | " " |

Table 13: Party Information for Proxy Agent

| | | | |
|----------|-------------------|--------------|---------------|
| Context | Proxy Destination | Proxy Source | Proxy Context |
| ducksoup | harpo | zeppo | bigstore |
| bigstore | groucho | chico | ducksoup |

Table 14: Proxy Relationships for Proxy Agent

| Target | Subject | Context | Privileges |
|---------|---------|----------|------------------------------|
| chico | groucho | ducksoup | 35 (Get, GetNext & GetBulk) |
| groucho | chico | ducksoup | 132 (Response & SNMPv2-Trap) |
| harpo | zeppo | bigstore | 35 (Get, GetNext & GetBulk) |
| zeppo | harpo | bigstore | 132 (Response & SNMPv2-Trap) |

Table 15: Access Information for Native Proxy

As represented in Table 13, the proxy agent party operates at UDP port 161 at IP address 1.2.3.5 using the party identity chico; the example manager operates at UDP port 2002 at IP address 1.2.3.4 using the identity groucho; the proxy source party operates at UDP port 161 at IP address 1.2.3.5 using the party identity zeppo; and, the proxy destination party operates at UDP port 161 at IP address 1.2.3.6 using the party identity harpo. Messages generated by all four SNMPv2 parties are authenticated as to origin and integrity by using the authentication protocol v2md5AuthProtocol and distinct, private authentication keys. Although these private authentication key values ("0123456789ABCDEF", "GHIJKL0123456789", "MNOPQR0123456789", and "STUVWX0123456789") are presented here for expository purposes, knowledge of private keys is not normally afforded to human beings and is confined to those portions of the protocol implementation that require it.

Table 14 shows the proxy relationships known to the proxy agent. In particular, the SNMPv2 context ducksoup refers to a relationship that is satisfied when the SNMPv2 party zeppo communicates with the SNMPv2 party harpo and references the SNMPv2 context bigstore.

In order to interrogate the proxied device associated with the party harpo, the management station groucho constructs a SNMPv2 GetNext request contained with a SnmpMgmtCom value which references the SNMPv2 context ducksoup, and transmits it to the party chico operating (see Table 11) at UDP port 161 and IP address 1.2.3.5. This request is authenticated using the private authentication key "0123456789ABCDEF".

When that request is received by the party chico, the originator of the message is verified as being the party groucho by using local knowledge (see Table 13) of the private

authentication key "0123456789ABCDEF". Because party groucho is authorized to issue GetNext (as well as Get and GetBulk) requests with respect to party chico and the SNMPv2 context ducksoup by the relevant access control policy (Table 15), the request is accepted. Because the local database of context information indicates that the SNMPv2 context ducksoup refers to a proxy relationship, the request is satisfied by its translation into a corresponding SNMPv2 GetNext request directed from party zeppo to party harpo referencing SNMPv2 context bigstore. This new communication is authenticated using the private authentication key "STUVWX0123456789" and transmitted to party harpo at the IP address 1.2.3.6.

When this new request is received by the party harpo, the originator of the message is verified as being the party zeppo by using local knowledge of the private authentication key "STUVWX0123456789". Because party zeppo is authorized to issue GetNext (as well as Get and GetBulk) requests with respect to party harpo and the SNMPv2 context bigstore by the relevant access control policy (Table 15), the request is accepted. A SNMPv2 Response message representing the results of the query is then generated by party harpo to party zeppo referencing SNMPv2 context bigstore. This response communication is authenticated as to origin and integrity using the private authentication key "MNOPQR0123456789" and transmitted to party zeppo at IP address 1.2.3.5 (the source address for the corresponding request).

When this response is received by party zeppo, the originator of the message is verified as being the party harpo by using local knowledge (see Table 13) of the private authentication key "MNOPQR0123456789". Because party harpo is authorized to issue Response communications with respect to party zeppo and SNMPv2 context bigstore by the relevant access control policy (Table 15), the response is accepted, and is used to construct a response to the original GetNext request, indicating a SNMPv2 context of ducksoup. This response, from party chico to party groucho, is authenticated as to origin and integrity using the private authentication key "GHIJKL0123456789" and is transmitted to the party groucho at IP address 1.2.3.4 (the source address for the original request).

When this response is received by the party groucho, the originator of the message is verified as being the party chico by using local knowledge (see Table 13) of the private

authentication key "GHIJKL0123456789". Because party chico is authorized to issue Response communications with respect to party groucho and SNMPv2 context ducksoup by the relevant access control policy (Table 15), the response is accepted, and the interrogation is complete.

4.5. Public Key Configuration

This section presents an example configuration predicated upon a hypothetical security protocol. This hypothetical protocol would be based on asymmetric (public key) cryptography as a means for providing data origin authentication (but not protection against disclosure). This example illustrates the consistency of the administrative model with public key technology, and the extension of the example to support protection against disclosure should be apparent.

| | | |
|---------------|--------------------|--------------------|
| Identity | ollie (agent) | stan (manager) |
| Domain | snmpUDPDomain | snmpUDPDomain |
| Address | 1.2.3.4, 161 | 1.2.3.5, 2004 |
| Auth Prot | pkAuthProtocol | pkAuthProtocol |
| Auth Priv Key | "0123456789ABCDEF" | " " |
| Auth Pub Key | "0123456789abcdef" | "ghijkl0123456789" |
| Auth Clock | 0 | 0 |
| Auth Lifetime | 300 | 300 |
| Priv Prot | noPriv | noPriv |
| Priv Priv Key | " " | " " |
| Priv Pub Key | " " | " " |

Table 16: Party Information for Public Key Agent

The example configuration comprises a single SNMPv2 agent that interacts with a single SNMPv2 management station. Tables 16 and 17 present information about SNMPv2 parties that is by the agent and manager, respectively, while Table 5 presents information about the local access policy that is known to both manager and agent.

| | | |
|---------------|--------------------|--------------------|
| Identity | ollie (agent) | stan (manager) |
| Domain | snmpUDPDomain | snmpUDPDomain |
| Address | 1.2.3.4, 161 | 1.2.3.5, 2004 |
| Auth Prot | pkAuthProtocol | pkAuthProtocol |
| Auth Priv Key | " " | "GHIJKL0123456789" |
| Auth Pub Key | "0123456789abcdef" | "ghijkl0123456789" |
| Auth Clock | 0 | 0 |
| Auth Lifetime | 300 | 300 |
| Priv Prot | noPriv | noPriv |
| Priv Priv Key | " " | " " |
| Priv Pub Key | " " | " " |

Table 17: Party Information for Public Key Management Station

As represented in Table 16, the example agent party operates at UDP port 161 at IP address 1.2.3.4 using the party identity ollie; the example manager operates at UDP port 2004 at IP address 1.2.3.5 using the identity stan. Both ollie and stan authenticate all messages that they generate as to origin and integrity by using the hypothetical SNMPv2 authentication protocol pkAuthProtocol and their distinct, private authentication keys. Although these private authentication key values ("0123456789ABCDEF" and "GHIJKL0123456789") are presented here for expository purposes, knowledge of private keys is not normally afforded to human beings and is confined to those portions of the protocol implementation that require it.

In most respects, the interaction between manager and agent in this configuration is almost identical to that in the example of the minimal, secure SNMPv2 agent described above. The most significant difference is that neither SNMPv2 party in the public key configuration has knowledge of the private key by which the other party authenticates its transmissions. Instead, for each received authenticated SNMPv2 communication, the identity of the originator is verified by applying an asymmetric cryptographic algorithm to the received message together with the public authentication key for the originating party. Thus, in this configuration, the agent knows the manager's public key ("ghijkl0123456789") but not its private key ("GHIJKL0123456789"); similarly, the manager knows the agent's public key ("0123456789abcdef") but not its

```
private key ("0123456789ABCDEF").
```

5. Security Considerations

In order to participate in the administrative model set forth in this memo, SNMPv2 implementations must support local, non-volatile storage of the local database of party information. Accordingly, every attempt has been made to minimize the amount of non-volatile storage required.

6. Acknowledgements

This document is based, almost entirely, on RFC 1351.

7. References

- [1] Case, J., Fedor, M., Schoffstall, M., Davin, J., "Simple Network Management Protocol", STD 15, RFC 1157, SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, May 1990.
- [2] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1448, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
- [3] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1442, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
- [4] McCloghrie, K., and Galvin, J., "Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1447, Hughes LAN Systems, Trusted Information Systems, April 1993.
- [5] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1449, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.
- [6] Galvin, J., and McCloghrie, K., "Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1446, Trusted Information Systems, Hughes LAN Systems, April 1993.
- [7] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1450, SNMP Research, Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, April 1993.

8. Authors' Addresses

James M. Galvin
Trusted Information Systems, Inc.
3060 Washington Road, Route 97
Glenwood, MD 21738

Phone: +1 301 854-6889
EMail: galvin@tis.com

Keith McCloghrie
Hughes LAN Systems
1225 Charleston Road
Mountain View, CA 94043
US

Phone: +1 415 966 7934
Email: kzm@hls.com

