**SNMP - RFC 3411 – Summarization**

**Introduction:**

Simple Network Management Protocol is a part of the internet protocol which is used for network management by collecting information from the devices which are connected to the network like printers, switches, routers etc and configuring those network devices. RFC 3411 describes about the architecture for the SNMP managed frameworks. The need for this is to make SNMP modular. This architecture majorly contains message processing, security, access control subsystems and multiple SNMP applications.

**SNMP:**

The system contains several nodes, each with command responder, agents, command generators, manager, management protocol. The main goal of this architecture is to make use of the existing protocols, keep it simple, SET support, make it cheaper. Some security threat that are applicable here are principal threats (alteration of the message in transit, masquerade) and secondary threats (malicious reordering of the messages, eavesdropping)[1].

**Elements of SNMP Frameworks Architecture:**

The following are the elements of the SNMP Frameworks Architecture.

1. Naming Based on Entities
2. Naming Based on Identities
3. Naming Based on Management Information

**Naming Based on Entities:**

Each entity comprises of applications and engine. These are explained below.

Engine: Each engine has unique ID, one to one association with the entity. It can be used to identify it within the administrative domain. Different administrative domains may have same ID. Each engine consists of the following:

1. Dispatcher: Each SNMP engine has one dispatcher. Each engine can have different versions of SNMP messages. Dispatcher supports this type of messages. PDU Dispatcher provides the following primitives
   1. To generate outgoing notification or request – sendPDU()
   2. To process an incoming PDU – processPDU()
   3. To generate outgoing response – returnReponsePDU()
   4. To process incoming response PDU- processResponsePDU()
2. Message processing subsystems: The main job of this subsystem is to prepare the message for sending and to extract the data from the received messages. It can have multiple models like SNMPv3, SNMPv2c, SNMPv1 etc on it. Each of these models takes care of the version specific formatting. The primitives provided by this subsystem are:
   1. Preparing an outgoing SNMP request or notification – prepareOutgoingMessage()
   2. Preparing an outgoing SNMP response message – prepareResponseMessage()
   3. Preparing data elements from incoming message – prepareDataElements()
3. Security subsystem: They contain multiple security models and the main job of this subsystem is to provide authentication and security services. This subsystem provides the following primitives:
   1. Generate notification message or request – generateRequestMsg()
   2. Process an incoming message – processIncomingMsg()
   3. Generate response message – generateResponseMsg()
4. Access control subsystem: This subsystem too provides authentication services, but this does it by means of access control models. This subsystem provides the following primitive:
   1. To check if the access is allowed – isAccessAllowed()

Applications: This consist of several applications like command generator, responder, notification originators, receivers and proxy forwarders. SNMP Engine provides services which are being used by these applications.

1. SNMP Manager: The SNMP Manager comprises of the notification receiver and/or the command generators. There could be more than one command generators in the SNMP Manager. Along with these it also contains its corresponding engine.
2. SNMP Agent: The SNMP Agent is similar to that of the SNMP Manager but with the following differences. This consist of command responder instead of the command generators and/or notification originator instead of the notification receiver.

**Naming Based on Identities:**

Principal: Principal is the one for whom the service is being provided. It could be anything. For example, it could be an individual, or group of individuals, application or group of application etc.

Security Name: This is in model independent format. This is used to represent the above-mentioned principal in a form which could be easily read by the humans.

Model Dependent Security ID: The security name which is mentioned above is independent of the model. But this security ID is specific to the particular security model.

**Naming Based on Management Information:**

This is located at SNMP Entity and uses a contextEngineID = snmpEngineID for its associated engine.

SNMP Context: SNMP Entity which could access the management information are collectively called as context. Hence, we can say that the SNMP Entity has access to many contexts. There will be a unique identification within every management domain. In an administrative domain, to uniquely identify a context, a combination of contextName and contextEngineID could be used.

Context Engine ID: To uniquely identify an SNMP entity in an administrative domain, we could make use of contextEngineID.

Context Name: Every SNMP Entity should have a unique contextName. This is used to for naming a context.

Scoped PDU: PDU, Context Name and Context Engine ID together forms a block which is called as scopedPDU. PDU – Protocol Data Unit is used to identify context by combining contextName and contextEngineID within an administrative domain.

**Security Consideration:**

This RFC describes how the security and access control model could be included during the implementation. But the level of security which could be achieved using this is based on the model which is implemented. It also suggests that the implementor should take care in protecting the data from disclosure.

**References:**

[1] - Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC 3411, DOI 10.17487/RFC3411, December 2002, <https://www.rfc-editor.org/info/rfc3411>.