A new mechanism for distributed managers persistence

Keywords: Network management, SNMP, Mobile agents, XML

Paper Area: 4 - Internet Computing and Electronic Commerce

Sub-topics: Agents for Internet Computing

Abstract

SNMP is currently a worldwide used network management framework. However, it only describes the structure of management information and how to access it, i.e., low-level operations, presenting no considerations about information interpretation or other high-level tasks.

Recent work in this area has managed to define several modules suitable for management distribution (DISMAN), which implies building agents that can cope with rather complex information structures. In the case of information loss, for example due to an agent reset, it is not practicable to force the management station to define and configure the information once again.

In this paper we present an XML based data model that can provide a structure for distributed managers persistency. Moreover, it can be used to define macros to group together elementary SNMP operations and to describe what should a mobile agent do when meeting some SNMP agent.

1 Introduction

The dependency of today's enterprises and business processes on networks infrastructure, leaves no space for downtime and failures, though infrequent. The current expectation for a robust network imposes that operational level is above 99.9% of the time. This fact encourages the evolution of network management mechanisms.

SNMP, the dominant network management architecture, is currently on its third version with more features and improved security mechanisms. During the last decade, this model has successfully generated more than 100 MIB modules with more than 10.000 objects, applicable to several management scenarios, from UPS monitoring to management distribution [1].

At the same time, the increasing on features and on complexity highlights further the lack on SNMP agents to provide some kind of persistency of well-functioning states. Stating a common wish on MIB modules: "… a robust, polite implementation would certainly not force its managing applications to reconfigure it whenever it resets".

The SNMP framework, and in particular MIB modules, requires creating or changing several objects values for the definition of a single task. This situation is similar to Assembly programming, where many instructions are necessary to define a single, higher-level operation.

Several approaches have been taken already to solve this limitation, most of them based on high level APIs that provide aggregation and correlation of simple SNMP operation[2][3][4]. However, there are numerous situations where a lower level persistence is desirable such as for a direct access to CLI devices, in the prototyping and development of SNMP entities or even as a database for policy management.

During the previous years we have been involved in the development, integration and evaluation of mobile agents in distributed management scenarios [5][6]. The work that we will present here constitutes the base for most of the results already obtained.

The following paragraphs show of XML can be used to describe SNMP agent objects, to define SNMP macros and to build guidelines that instruct mobile agents how to act near SNMP information. It will be described yet how this methodology follows recent proposals inside IETF that suggest the replacement of SMI by XML in the definition of management information.

2 XML for SNMP operation description

The *Structure for Management Information* guidelines (SMI) describe the structures and schemes for the definition of management information but do not describe SNMP commands or messages PDU [7]. In a more powerful way, XML can be used not only to exchange SMI information [8], but also to describe SNMP commands.

SNMP defines eight SNMP commands and the associated PDUs [9]: *get-request*, *get-next-request*, *get-bulk-request*, *response*, *set-request*, *inform-request*, *snmpV2-trap* and *report*, although the usage and precise semantics of this last PDU are not presently defined. According to the role that the SNMP entity takes, it may use different commands (Figure 1).

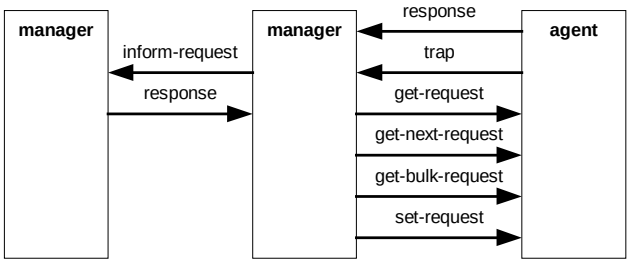

Figure 1 – Command exchange between SNMP entities.

All the PDUs share the same fields except for the get-*bulk-request*, which has identical structure but differs in two fields (Figure 2).

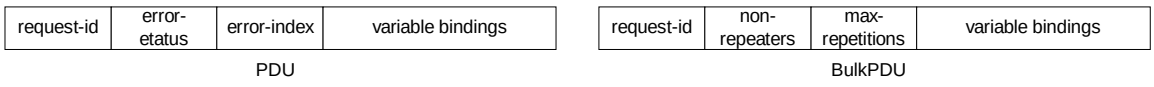

Figure 2 – SNMP PDUs.

The field *request-id* is used by SNMP applications to distinguish requests and to respond accordingly. If *error-status* is non-zero, it means that some error occurred, and in this case, *error-index* must indicate the variable binding that caused the exception.

BulkPDUs although use the same structure but with different fields. The *non-repeaters* (N) field indicates the number of variable bindings starting at the beginning that should return a single value. This is retrieved by *get-next* like operation. For each of the remaining variable bindings, *max-repetitions* (M) values are returned following the *get-next* operation. This means that the response to a *get-bulk-request* may have N+(M*R) variable bindings, where R is the remaining number of variable bindings, i.e., the total number of variable bindings minus N.

In summary, there are several kinds of SNMP commands with a number of variable bindings. Each sequence of commands makes an operation, or task. Several tasks can be grouped as an SNMP document. The Extensible Markup Language (XML) allows describing structured information by defining specific tags [10]. This tag arrangement builds a document that can be used to exchange information independently of the platform, programming language or application objective. These characteristics make it ideal for representing SNMP commands, tasks or management operations. This means defining a tag for document root including sufficient information for protocol operation such as security parameters and others. An example can be:

```
1. <snmp version=”3” user=”senior” authProtocol=”MD5” authPassword=”senior”
privPassword=”senior”>
2.  <mib name=”RFC1213-MIB” location=”file:/usr/local/mibs”/>
3.
4.  <property name=”trapObject” value=”.1.3.6.1.2.4.5.6”/>
5.  <property name=”otherObject” value=”.1.3.6.1.2.4.5.7”/>
6.  <property name=”destination” value=”192.168.168.168:162”/>
7.
8.  <task name=”myTrap”>
9.    <trap version=”2” destination=”$destination”>
10.      <!—implicit VarBinds
11.      <varBind name=”sysUpTime” oid=”.1.3.6.1.2.1.1.3.0”/>
12.      <varBind name=”snmpTrapOID” oid=”.1.3.6.1.6.3.1.1.4.1”/>
13.      -->
14.      <varBind name=”someOID” oid=”$trapObject”/>
15.    </trap>
16.
17.    <trap version=”2” destination=”$destination” name=”someOID” oid=”$otherObject”/>
18. </task>
19.
20. <runTask name=”someTask” document=”file:/usr/local/operation/someOperation.xml”/>
21.
22. </snmp>
```

Each SNMP document may use several MIB modules that should be loaded and parsed before initiating the operations, so this information may be included with the tag `<mib>`.

The `<property>` tag permits defining parameters in the XML document, thus allowing using the same code for different situations. The property name prefixed by a dollar sign is then replaced by the value defined in the correspondent `<property>` tag (line 9).

Each task contains several commands and provides a name for the group. Besides trap, the commands can be:

```
<get>…</get>
<getNext>…</getNext>
<getBulk nonrep=”N” maxrep=”M”>…</getBulk>
<set>…</set>
<inform>…</inform>
<response>…</response>
```

Command payload, or the variable binding list, is defined as a sequence of `<varBind>` tags with three parameters: `name`, `oid` and `value`. The value parameter can be omitted if there is none.

For commodity, when a command has a single variable binding it can be represented in short notation (line 17), thus replacing the larger form (line 9).

It is also important to provide a mechanism that allows including previously defined tasks. This alleviates the burden of repeating in the same document operations defined somewhere else (line 20).

3 Usage scenarios

There are several scenarios where this approach can be used. Agents use it to save objects values or to synchronize information with other agents, possibly from other vendors. Management applications may use it to automate a complex series of operations by grouping them together as a single task.

Mobile agents are emergent technology that can provide a valuable assistance in the conception of distributed systems, particularly for network management applications. However, the domination of SNMP management framework in this area suggests that an interaction between mobile agents and SNMP entities will occur. The interaction will work in both directions, i.e., managing mobile agents through SNMP [4] and monitoring and controlling SNMP agent by using mobile tools [11]. The latter requires that the mobile agent knows what to do near each SNMP agent, also a possible usage scenario for SNMP operation description in XML.

3.1 Agent persistency

SNMP agents may have a huge number of stored values and some may be consequence of *set-request* operations issued by the manager. The Event MIB [12], for example, has several tables that are set by the manager to define trigger conditions, events, notifications and actions. If the agent has a failure, for some reason, these values are lost. In this case, if the agent does not have some persistency mechanism the manager must issue them once again. For this reason, it is important that agents are capable of storing its’ objects values regardless of the available medium – local disk, network file system or other (Figure 3).

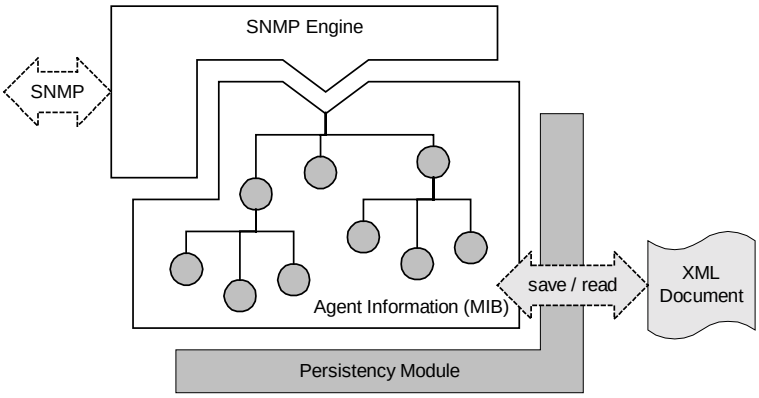

Figure 3 – SNMP agent persistency.

MIBs usually do not have explicit control for the persistence of values and it may be very different according to the agent manufacturer and other implementation details, such as the programming language.

XML can be used to explicitly control how the objects should be created after an agent disruption, regardless of the medium or programming language. Moreover, it allows agents to replicate data to other agents, a important issue in management delegation.

3.2 SNMP Macros

A macro is a series of commands and instructions grouped together as a single command to accomplish a task automatically. Instead of manually performing a series of time-consuming, repetitive actions, it is possible to create and run a single macro — in effect, a custom command — that accomplishes the task.

On a practical scenario, the user interacts with a graphical user interface to create a set of macro templates suitable for the desired operations. These macros can then be stored for future use (Figure 4).

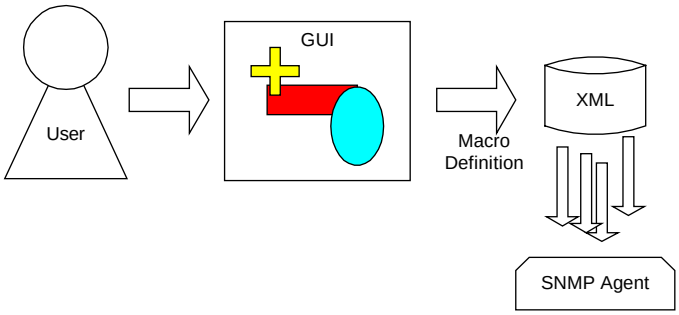

Figure 4 – SNMP macro definition.

When executing the macro, the management system reads the XML document and performs the described operations in sequence. As an example, the creation of the expression:

$$\text{utilization} = \frac{(\text{ifInOctets} + \text{ifOutOctets}) * 800}{\text{seconds} * \text{ifSpeed}},$$

which indicates the line utilization rate on a DISMAN Expression MIB module agent [13] requires setting 32 objects [14].

By using SNMP macros in XML, it is possible to define one task for the conditional, hard-limit expression with two *set-request*s and another task for the utilization expression with five *set-request*s. Each command variable binding provides the necessary information to write or to create the correspondent agent object. The following example shows this procedure.

```
<snmp community="public" writeCommunity="private">
  <mib name="DISMAN-EXPRESSION-MIB" location="file:/usr/local/mibs"/>
  <mib name="RFC1213-MIB" location="file:/usr/local/mibs"/>

  <task name="hard_limit">
    <set>
      <varbind name="expExpression" oid="expExpression.2.'me'.4.'hard'" value="$1==1"/>
      <varbind name="expExpressionValueType" oid="expExpressionValueType.2.'me'.4.'hard'"
value="2"/>
      <varbind name="expExpressionRowStatus" oid="expExpressionRowStatus.2.'me'.4.'hard'"
value="1"/>
    </set>

    <set>
      <varbind name="expObjectID" oid="expObjectID.2.'me'.4.'hard'.1"
value="ifConnectorPresent"/>
      <varbind name="expObjectWildcard" oid="expObjectWildcard.2.'me'.4.'hard'.1"
value="1"/>
      <varbind name="expObjectSampleType" oid="expObjectSampleType.2.'me'.4.'hard'.1"
value="1"/>
      <varbind name="expObjectRowStatus" oid="expObjectID.2.'me'.4.'hard'.1"
value="ifConnectorPresent"/>
    </set>
  </task>

  <task name="line_utilization">
    <set>
      <varbind name="expExpression" oid="expExpression.2.'me'.4.'util'"
value="($1+$2)*800/$4/$3"/>
      <varbind name="expExpressionValueType" oid="expExpressionValueType.2.'me'.4.'util'"
value="integer32"/>
      <varbind name="expExpressionDeltaInterval"
oid="expExpressionDeltaInterval.2.'me'.4.'util'" value="6"/>
      <varbind name="expExpressionRowStatus" oid="expExpressionRowStatus.2.'me'.4.'util'"
value="1"/>
    </set>

    <set>
      <varbind name="expObjectID1" oid="expObjectID.2.'me'.4.'util'.1"
value="ifInOctets"/>
      <varbind name="expObjectWildcard1" oid="expObjectWildcard.2.'me'.4.'util'.1"
value="1"/>
      <varbind name="expObjectSampleType1" oid="expObjectSampleType.2.'me'.4.'util'.1"
value="2"/>
      <varbind name="expObjectConditional1" oid="expObjectConditional.2.'me'.4.'util'.1"
value=" expValueUnsigned32Val.4.'hard'.0.0"/>
      <varbind name="expObjectConditionalWildcard1"
oid="expObjectConditionalWildcard.2.'me'.4.'util'.1"  value="1"/>
      <varbind name="expObjectDiscontinuityID1"
oid="expObjectDiscontinuityID.2.'me'.4.'util'.1"  value="ifCounterDiscontinuityTime"/>
      <varbind name="expObjectDiscontinuityIDWildcard1"
oid="expObjectDiscontinuityIDWildcard.2.'me'.4.'util'.1"  value="1"/>
      <varbind name="expObjectRowStatus1" oid="expObjectRowStatus.2.'me'.4.'util'.1"
value="1"/>
    </set>

    <set>
```

```
      <varbind name="expObjectID2" oid="expObjectID.2.'me'.4.'util'.2"
value="ifOutOctets"/>
      <varbind name="expObjectWildcard2" oid="expObjectWildcard.2.'me'.4.'util'.2"
value="1"/>
      <varbind name="expObjectSampleType2" oid="expObjectSampleType.2.'me'.4.'util'.2"
value="2"/>
      <varbind name="expObjectRowStatus2" oid="expObjectRowStatus.2.'me'.4.'util'.2"
value="1"/>
    </set>

    <set>
      <varbind name="expObjectID3" oid="expObjectID.2.'me'.4.'util'.3"  value="ifSpeed"/>
      <varbind name="expObjectWildcard3" oid="expObjectWildcard.2.'me'.4.'util'.3"
value="1"/>
      <varbind name="expObjectSampleType3" oid="expObjectSampleType.2.'me'.4.'util'.3"
value="1"/>
      <varbind name="expObjectRowStatus3" oid="expObjectRowStatus.2.'me'.4.'util'.3"
value="1"/>
    </set>

    <set>
      <varbind name="expObjectID4" oid="expObjectID.2.'me'.4.'util'.4"
value="sysUpTime.0"/>
      <varbind name="expObjectWildcard4" oid="expObjectWildcard.2.'me'.4.'util'.4"
value="2"/>
      <varbind name="expObjectSampleType4" oid="expObjectSampleType.2.'me'.4.'util'.4"
value="2"/>
      <varbind name="expObjectRowStatus4" oid="expObjectRowStatus.2.'me'.4.'util'.4"
value="1"/>
    </set>

  </task>
</snmp>
```

3.3 Mobile guidelines for SNMP operations

The introduction of other kind of technology in management models may help reduce insufficiencies or improve the way it works, such as reducing network load or improving offline operation. One of such technology is mobile agent, which can be used as location independent management tools and carrying the instructions to perform near SNMP agents [11].

Mobile network agents are pieces of code that can be dispatched from one computer and transported to a remote computer for execution. Arriving at the remote computer, they present their credentials and obtain access to local services and data (maintaining the previous execution state). The remote computer may also host SNMP agents thus providing a place for local interaction between them.

Nowadays there are several kinds of technological solutions for the development of mobile agent based systems. All of them provide a platform to support the migration and execution of code (agent platform) and use scripting or interpreted languages (such as TCL or Java) to survive over heterogeneity [15].

The manager must program mobile agents with the sequence of task to perform when meeting an SNMP agent. The interaction between both agents must use SNMP commands, to update or to retrieve information. This scenario can be seen as mobile macro execution and, although the proposed syntax does not provide intelligence, may save bandwidth.

Higher-level operations could be defined, for example, searching for a value in an SNMP table or waiting for some condition to occur. These higher-level operations would use SNMP tasks that can be described through the XML proposed syntax.

```
<snmp community="public" writeCommunity="private">
  <task name="exampleOperation">
    <get name="sysDescr" oid=".1.3.6.1.2.1.1.1.0"/>
    <getNext>
      <varBind name="ifTable" oid=".1.3.6.1.2.1.2.2"/>
      <varBind name="otherObject" oid=".1.3.6.1.2.3"/>
    </getNext>
    <set>
      <varBind name="expExpression" oid=".1.3.6.1.2.1.90.1.2.1.1.3.1.100.1.101"
value="$1*10"/>
    </set>
    <getBulk nonrep="1" maxrep="50">
      <varBind name="expExpression" oid=".1.3.6.1.2.1.90.1.2.1.1.3"/>
      <varBind name="expValue" oid=".1.3.6.1.2.1.90.1.3.1"/>
    </getBulk>
  </task>
</snmp>
```

4 Conclusions

XML documents provide an ideal arrangement for describe structured information, for example SMI such as suggested by the ongoing work on the IETF, or for specify sequences of SNMP operations such as was presented in this paper. This later approach can provide a method to reinstate distributed managers data lost due to planned or unexpected resets. Storing this kind of documents, either on disc or by using other mechanism, provide a persistency mechanism for such agents, which can read it when reinitiating.

Alternatively, the same structure allows grouping commands and instructions together as single commands to accomplish tasks automatically. This would allow building a library of sets of basic SNMP commands, or macros, thus providing higher-level operations to the user.

Several researchers and industry see mobile agents as a promising technology for network management. Nowadays, they are helpful if they manage to interact with existing SNMP agents and management applications. The use of XML documents with sets of SNMP commands can also be used to describe its behaviour near SNMP agents.

5 References

[1] J. Postel, J. Reynolds, "Internet Official Protocol Standards", STD 1, September 1998.

[2] J. L. Oliveira, J. A. Martins, "A Management Architecture based on Network Topology Information", *Journal of Network and Systems Management*, Vol. 2, Nº 4, pg. 401-414, Dezembro 1994.

[3] G. Goldszmidt, Y. Yemini, "Delegated Agents for Network Management", *IEEE Communications Magazine*, Vol. 36 No. 3, pgs. 66-71, March 1998.

[4] R. Lopes, J. Oliveira, "Managing Mobile Agents with SNMP", *Proc. of the Simposio Español de Informatica Distribuida* – SEID 2000, September 2000, Orense, Spain.

[5] J. Oliveira, R. Lopes, “Distributed Management based on Mobile Agents”, *Proc. of the 1st International Workshop on Mobile Agents for Telecommunications Applications* – MATA’99, October 1999, Ottawa, Canada.

[6] R. Lopes, J. Oliveira, “SNMP Management of MASIF Platforms”, IFIP/IEEE International Symposium on Integrated Management 2001 – IM2001, May 2001, Seattle, USA.

[7] K. McCloghrie, D. Perkins, J. Schoenwaelder, J. Case, K. McCloghrie, M. Rose, S. Waldbusser, “Structure of Management Information Version 2 (SMIv2)”, RFC2578, April 1999.

[8] J. Schoenwaelder, F. Strauss, “Using XML to Exchange SMI Definitions”, Internet Draft draft-irtf-nmrg-smi-xml-00.txt, June 2000.

[9] J. Case, K. McCloghrie, M. Rose, S. Waldbusser, “Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)”, RFC1905, January 1996.

[10] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0", W3C REC-xml-19980210, February 1998 (http://www.w3.org/TR/1998/REC-xml-19980210).

[11] R. Lopes, J. Oliveira, “Distributed Management of Mobile Components”, *ConfTele2001 - 3ª Conferência de Telecomunicações*, Figueira da Foz, Portugal, April 2001.

[12] R. Kavasseri, B. Stewart, “Event MIB”, RFC2981, October 2000.

[13] R. Lopes, J. Oliveira, “Distributed Management: Implementation issues”, *proc. of the International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet – SSGRR2000*, l’Aquila, Rome, Italy, August 2000.

[14] R. Kavasseri, B. Stewart, “Distributed Management Expression MIB”, RFC2982, October 2000.

[15] R. P. Lopes, J. L. Oliveira, “Software Agents in Network Management”, *Proc. of the 1st International Conference on Enterprise Information Systems – ICEIS’99*, March 1999, Setubal, Portugal.