

Script MIB Extension for Resource Limitation in SNMP Distributed Management Environments

Atslands da Rocha¹, Cris Amon da Rocha², and J. Neuman de Souza¹

¹ Universidade Federal do Ceará (UFC) - Dept. de TeleInformática (DETI)
Campus do Pici - Fortaleza - Brazil

atslands@deti.ufc.br, neuman@ufc.br

² Faculdade 7 de Setembro (FA7)

Alm. Maximiano da Fonseca 1395, Luciano Cavalcante - Fortaleza - Brazil
crisamon@fa7.edu.br

Abstract. Resource limitation on management scripts has been an important requirement for distributed management environments. This article proposes a Script MIB extension with new objects able to control the usage of specific resources (physical memory, processing cycles, among others) for each script launched inside the distributed environment.

1 Introduction

A widespread dissemination of TCP/IP networks has caused an increasing search for new and more efficient management environments. Following this purpose the Internet Engineering Task Force (IETF) has created the Simple Network Management Protocol (SNMP), which supplies a functional management framework.

However, the traditional SNMP management model is becoming a problem because of processing bottlenecks and bandwidth saturation. In large networks this kind of management environment is impracticable and opens room for new frameworks like distributed management model.

Development of distributed management standards for SNMP environments is a task for DISMAN-WG [1] that has released a set of new Management Information Bases (MIB) and distributed management framework documents to improve the SNMP model. The main purpose is to avoid changes in the protocol and make use of the actual devices already installed.

One of these proposals uses management by delegation [2], where a high level manager is responsible for submanagers. This proposal defines management functions by means of scripts and makes use of the Script MIB to provide mechanisms for the transfer, execution, administration and control of management scripts.

With this kind of environment, problems of bottlenecks and bandwidth saturation are solved by distribution of management tasks. Problems like scalability and fault tolerance are covered too. In spite of these advantages, it's necessary and important to analyze tradeoffs for submanagers. These elements are network devices with different functions and, therefore, different resources (Processor, memory and bandwidth). These functions cannot be damaged by scripts.

Aware of the problem cited above, DISMAN advises hardware resource limitation in management scripts execution, but anything else is specified. Thus, this article proposes a Script MIB extension in order to deploy this requirement. New management objects are added in order to control hardware resources in distributed managers.

2 IETF Script MIB

The IETF Script MIB provides means of delegating and calling management scripts for/in distributed managers. According to RFC 3165 [3], a distributed manager is a processor entity able to provide network management functions. This distributed manager can be broken up in two elements. The first one is a SNMP entity, which implements the Script MIB, and the other is an execution environment responsible for execution of scripts.

A standard interface is defined by the SNMP management architecture for delegation of scripts. In short, the Script MIB provides mechanisms to: transfer management scripts to distributed manager; start, suspend, resume and finish management scripts; send arguments to management scripts; monitor/control active management scripts and bring management scripts execution results back.

Scripts can be written in any programming language supported by the MIB implementation. Nothing is said about the format of management scripts during transfers. The most part of programming languages is registered by IANA and other specific languages can be registered in enterprise subtrees.

The Script MIB allows the execution of various instances of the same scripts. A running script can be suspended, resumed or aborted. Active scripts timeouts are used to avoid problems such as infinite loops. A notification with an exit code, an error description and a timestamp is generated when a script ends in an abnormal way.

To start a script is needed to create an entry in smLaunchTable table, which store all script parameters, its maximum time to live and its owner. An abstract launch button is responsible for starting the script and this action can be done through a single SetRequest primitive. So many instances of the same script can be executed, but different parameters and permissions must be applied.

The smRunTable table lists all scripts that are running at the moment or finished one second before. A manager can retrieve script states, control their execution and retrieve results of finished scripts through objects from this table. The agent stores script results until maximum time to live.

However, the Script MIB specification leaves open the resource limitation problem. This issue is addressed just one time when suggestions of mapping the owner Script MIB object to a local operating system (or another execution environment) user are done.

3 Script MIB Extension

Mapping of the owner MIB object to a local user in the execution environment is a great suggestion to solve the resource limitation problem, however this approach is followed by many tradeoffs to be considered. The lack of user based policies in simple execution environments or no related information about users and their pre-defined resources are some examples. A user in an general purpose operating system nothing says about constraints such as limit of processing cycles.

Therefore, this article proposes the creation of a new table called smResourceTable, which provides new objects able to specify and control resources during management script executions. Some computational resources are very important to the right working of devices. Four essential features were chosen to control script execution: Processing time; Physical memory; Number of open files; File size. The Script MIB extended with the new table is shown at Fig. 1.

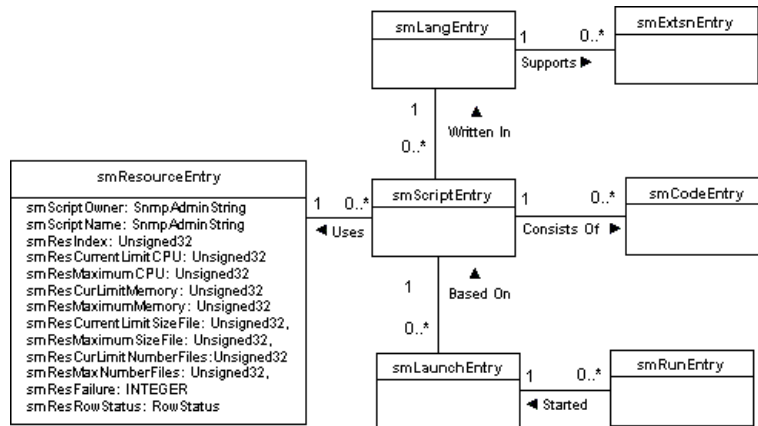


Fig. 1. Extended Script MIB diagram

An entry is associated with a management script through the smScriptOwner and smScriptName objects. smResourceIndex object associates a specific resource limitation configuration with a script execution. All attributes are optional.

smResourceCurrentLimitCPU object stores the maximum of processing time, in seconds, that a script can make use. smResourceCurrentLimitMemory stores the maximum physical memory available to its execution. Management scripts cannot exceed these limits and the execution environment needs certify this constraint.

Maximum limits that a user can setup to the current limit of processing time and physical memory are presented at smResourceMaximumCPU and sm-

ResourceMaximumMemory objects. Moreover, they cannot be changed by the script owner because are pre-defined by the SNMP agent.

The smResourceCurrentLimitFileSize object shows the maximum size, in bytes, of files written by management scripts and its maximum possible value is found at smResourceMaximumFileSize. In the same way, the smResourceCurrentLimitNumberFiles and smResourceMaximumNumberFiles objects store the maximum number of open files that a specific script can handle and the maximum possible values. Management scripts cannot exceed these values.

The smResourceFailure object points out that the related script has reached one of the pre-defined limits. This objects value is related with the last incident. Possible values are: sigxcpu (Processing time limit has been reached); notAllocMemory (Script has tried to exceed the maximum physical memory limit); sigxfsz (Script has tried to write to a file bigger than the maximum file size value defined); noOpenFile (Script has tried to open more files that the value defined); noFail (No limit has been reached yet).

When a script reaches a pre-defined limit for any resource or tries to exceed this limit, a notification is generated in order to identify which limit was responsible for the exception.

Even with definitions for each object cited above, there are so many ways to deploy them. Their behavior is left open for each environment that implements the proposed extension. However, two situations need to be considered:

The first one concerns about objects that store maximum possible values. Management scripts can belong to different owners, which can have different permissions and privileges in the system, so the agent responsible for the Script MIB can specify different values in order to give different priorities to the owners.

The second one is about penalties for management scripts that tried to exceed limits pre-defined at current objects. Notifications send to the manager are just warning messages and not penalties.

A complete Script MIB specification extended by the proposal cited in this article can be found at the project website [4].

4 Script MIB Extension Implementation

4.1 JASMIN - A Java Script MIB Implementation

The Script MIB has been implemented in Java programming language at Technical University of Braunschweig and C&C Research Laboratories of NEC Europe Ltd. at Berlin inside a project called JAvA Script Mib Implementation (JASMIN) [5]. The final projects purpose is to experiment the IETF Script MIB functionalities and share results with DISMAN working group. Figure 2 details the Jasmin internal structure.

The left side shows the master agent process, where Jasmin makes use of NET-SNMP toolkit . Subagents are loaded inside the master agent in runtime mode through dynamically loaded subagents modules.

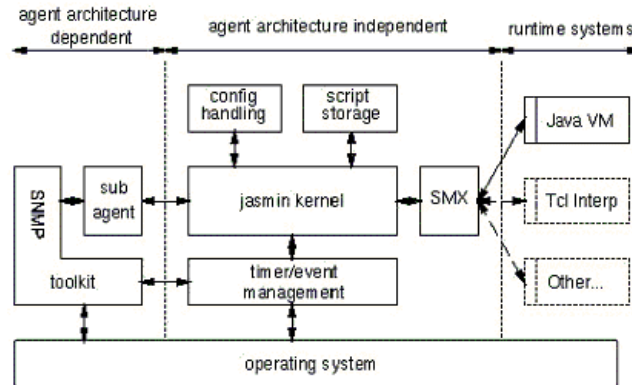


Fig. 2. JASMIN internal structure

Jasmin core shows the autonomous kernel and its auxiliaries mechanisms. This kernel changes information with execution systems via Script Mib eXtensibility (SMX) [6], which eases the adding of new execution systems. The Java virtual machine is used as the main Jasmin execution system and provides support for execution systems in Tcl and Perl. The Jasmin kernel starts execution systems and a local security service certifies the system authentication process.

Jasmin uses security profiles in order to define what an active script can do. The first one is the operating system security profile, which specifies a set of operating system services that can be used by the script. The second one is the runtime system security profile, which specifies a set of services that can be used at some well-defined moments during script execution.

4.2 The SmResourceTable Implementation

Jasmin is an open source SNMP agent built over a modular architecture that provides extensibility and scalability. Besides, it provides support for resource limitation in distributed managers.

Based on these remarks, Jasmin project has been chosen as a framework to the Script MIB extension implementation, following suggestions found at own Jasmin documentation of mapping script credentials to Jasmin security profiles.

Jasmin has three essential files: a MIB file that defines the module itself; a C header file with all function prototypes and C implementation files. The MIB file was generated according to RFC 2592 [7] and the C header file is called jasmin.h.

To add the new proposed objects to Jasmin environment it was needed change the local Script MIB file in order to extend it. Jasmin.h file was changed with new object definitions and their handle SNMP functions. Two files called resource.c and write_resourcetab.c have been added and concerns about read-only and read-write access methods.

Initial object values were specified in `jasmin.conf` configuration file, which has been changed with new `smResourceTable` objects. Another configuration file named `snmpd_jasmin.conf` was changed to support access to `smResourceTable`.

All new objects were implemented, but `smResourceCurrentLimitMemory` and `smResourceMaximumMemory` objects will not be instrumented because Jasmin haven't any limitation rule for them.

Object instrumentation was done at C language because of NET-SNMP toolkit. However, the resources limitation mechanism has been developed in Java. Warnings and `JasminSecurityException` exceptions are sent when a script tries to access an unauthorized resource. This implementation has a main purpose of validating the proposal presented in this article.

5 Final Remarks

Script MIB has many advantages in distributed network management. However, the possibility of execute management scripts in remote devices can bring many problems as controlling of submanagers; monitoring of scripts and their instances and controlling of arguments and scripts results. The computational processing to realize all these tasks cited above can damage essential distributed manager functions.

In order to define a model for delegation of credentials over the Script MIB, allowing resource limitation per script, this article presented a Script MIB extension, which add new objects to monitor and control essential system resources (processing cycles, physical memory and file handle).

The implementation has been finished and was deployed inside the JASMIN architecture, which has saved days of work. Some experiments will be done next step in order to certify viability and efficiency.

As future works, our group will try to port this extension for other execution environments such as Tcl and Perl, which couldn't be supported at this first moment. Besides, new experiments and improvements will be done in order to reach an IETF draft style as a RFC 3165 addendum.

References

1. DISMAN - The Distributed Management Working Group. Available at: <http://www.ietf.org/html.charters/disman-charter.html/>.
2. J. Schönwälder. Network Management by Delegation - From Research Prototypes Towards Standards. Computer Networks and ISDN Systems, Nov 1997.
3. D. Levi and J. Schönwälder. Definitions of Managed Objects for the Delegation of Management Scripts. RFC 3165, Nortel Networks, TU Braunschweig, Aug 2001.
4. XScript MIB Project. Available at: <http://www.deti.ufc.br/~atslands/xscriptmib>.
5. Jasmin - JAVa Script Mib ImplementatioN. Available at: <http://www.ibr.cs.tu.bs.de/projects/jasmin>.
6. J. Schönwälder, M. Bolz, S. Mertens. SMX Script MIB Extensibility Protocol Version 1.0. RFC 2593, Nortel Networks, TU Braunschweig, Aug 2001.
7. D. Levi and J. Schönwälder. Definitions of Managed Objects for the Delegation of Management Scripts. RFC 2592, Nortel Networks, TU Braunschweig, May 1999.