A Uniform Resource Identifier Scheme for SNMP

Rui Pedro Lopes
Polytechnic Institute of Bragança, ESTiG
5300 Bragança, Portugal
rlopes@ipb.pt

José Luis Oliveira
University of Aveiro, DET
3810 Aveiro, Portugal
jlo@det.ua.pt

***Abstract* – One of the World Wide Web characteristics, besides its omnipresence in computer systems, is the adoption of a universal user interface that is used to access several different services that were previously accessed individually by independent applications. The Internet resources started to be identified by URI schemes, a text string with specific syntax and grammar. Although existing for several services such as http, ftp, gopher and news, these identifiers are not used to identify SNMP resources.**

This paper proposes an URI scheme for identifying SNMP resources and presents some practical scenarios where the existence of such compact and complete identifying mechanism increases flexibility and functionality of network management applications.

Keywords: SNMP, URI, URL, Network Management.

I. INTRODUCTION

The information, services or any other kind of resource that surface daily on the Internet has an associated naming space partially based on DNS. The resource identification is performed through text strings known as URI (*Uniform Resource Identifiers*) [1]. This string has all the necessary information to identify, to retrieve or, eventually, to update the resource configuration.

Resources may have a physical nature, such as a processor, memory or storage devices, or a logical type, such as Web pages or network management agents. The resources are identified through references that reveal its address and nature. For example, an user mailbox is identified according to the scheme `mailto:<name>@<address>`.

Despite the intrinsic differences that allow to catalogue different resources, each identification follows common concepts such as the resource name and address. This set of characteristics allows using a common syntax to locate them regardless of its nature.

The uniform representation of references allows using identifiers for different resources in a common context. In the Internet, for example, the resources FTP, HTTP or NEWS are accessible by a common tool – the Internet browser. The resource is specified in the address field by the URI and, according to its grammar, it calls the appropriate tool for its processing and presentation.

Network and systems management has been largely dependent on the SNMP framework [2] in which a typically centralized management station is used to deal with the information retrieved from, or delivered to, SNMP agents (the resources modeling). SNMP services are typically made available through specially designed APIs, which depend on a set of arguments for management information identification and on the required action. Besides the problem of different APIs supporting different and incompatible interfaces, the distinct versions of the SNMP usually require diverse parameters. Such as in the Internet situation, it is useful to integrate the tools, paradigms and models around a common context.

In this paper we suggest using a specific URI scheme for SNMP thus solving the parameters dispersion associated to management operations. According to this view, the SNMP management may be based on a common tool, where different resources and operations are identified through particular URI semantics.

The authors are using the SNMP URI scheme as an informal tool (in the graphical user interfaces) for the management of mobile agent platforms. Proprietary tools, standard CORBA interfaces or SNMP are the usually mechanisms for this goal. However, to deal with all these different access schemes, we are using specific URLs to differentiate them on a common context [3].

Recently, two more projects where found to share the same concept. One of them, *iosnmp* [4], is a plug-in to the K desktop environment [5] that, after registration, allows using the *konqueror* browser as a MIB browser along with the file system or HTML pages. This plug-in accepts URI schemes such as `snmp://v3user@host:port/initialMibNode/` and supports only SNMPv3. At the security level it assumes the `authNoPriv` solution. Other parameters, such as the timeout and retries are not defined.

The *Cricket* application is another example and it consists of a resource monitoring tool. It is configured by specific

URLs, such as `snmp://community@host:port/source`, where the source stands for a managed object [6]. It supports only the SNMPv1 e SNMPv2c versions and like the previous example, it does not accept other parameters, such as timeout, retries or others.

Despite individual limitations, these examples support the need for the existence of URLs to describe SNMP resources regardless of version, security parameters or services. This universal syntax can be used transparently across different applications.

On the other hand, these three examples [3][4][6] are exclusively based in the manager side to provide access to management information. In this paper we seek to extend the SNMP URI formalism to the agent side as an identification mechanism more powerful than the OID (Object Identifier) to locate managed objects.

The paper structure is the following. Section II describes the syntax and grammar to identify SNMP resources in an URI compatible form which we designate SNMP URL. In section III we present some practical scenarios. The first introduces an URL-TARGET-MIB an alternative to the SNMP-TARGET-MIB. The second shows how a SNMP URL may be used to extend the Expression MIB [7] to accept remote expression parameters, currently in research. The third shows how it can be used to manage mobile agents [3].

II. A Uniform Resource Identifier for SNMP

The URI concept was defined by the networks working group of the IETF (*Internet Engineering Task Force*) as a universal specification for physical or logical resources [1]. It has a generic syntax, suitable for identifying a broad set of resources namely web pages, email addresses, newsgroups and books, among many others. Currently IANA has already registered 36 URI schemes for as much services [8].

A. *URI generic syntax*

At the first and higher level an URI scheme is as follows:

`[scheme:]scheme-specific-part[#fragment]`

The symbols '[' and ']' are delimiting optional sections and the symbols ':' e '#' stand for themselves and separate different URI sections. An URI is absolute when its schema exists and relative otherwise. A relative URI depends on an absolute URI to get the missing information, such as the scheme.

URIs may also be classified as a hierarchy. In this case, the scheme specific part starts with the symbol '/' and has the following format:

`[scheme:][//authority][path][?query][#fragment]`

The authority may be represented according to a hierarchical naming scheme (e.g.`[user-info@]host[:port]`). The `path` contains information to identify a given resource under the authority context. The section `query` has the information to be passed to and processed by the resource. Finally, the `fragment` has additional information to be used on the client side after the operation completes successfully. Theoretically, it does not belong to the URI since it is not used in the communication but it is frequently associated to it (for HTML bookmarks, for example).

The URI urn:isbn:096139210x, which corresponds to a book, and mailto:pemz@mail.hosting.pt, which refers to an email address, are non-hierarchical, or opaque according to the standard taxonomy.

The URI http://www.ics.uci.edu/pub/ietf/uri/#Related is both hierarchical and absolute because it has the scheme http and the scheme specific part starts with the symbol '/'. The authority represents an Internet name (www.ics.uci.edu) and the path identifies the resource /pub/ietf/uri/, unique in this server. The fragment "Related" is only used by the Internet browser to automatically show the HTML page at this position.

TABLE I. Some URI examples.

Model	URI
HTTP	http://www.det.ua.pt
FTP	ftp://jprs@ftp.univ.pt/private/projectX/
XMLORG	urn:xmlorg:objects:schema:xmlschema:xcatalog
NFS	nfs://server/a/b
LDAP	ldap://ldap.itd.umich.edu/o=U%20of%20Mich,c=US
MAIL	mailto:rlopes@ipb.pt

B. *SNMP parameters*

The functionality associated with SNMP entities, according to the SNMPv3 standard, is defined by grouping applications. These may be of five kinds: command generators, notification originators, command responders, notification receivers and proxy forwarders. A management agent is thus based on a command responder and a notification originator while a management station contains a command generator and a notification receiver.

SNMP entities are identified by a number – the `snmpEngineID`. Each entity may have several contexts, unique for that entity. To identify each managed object they are necessary four parameters: the SNMP engine identifier (`snmpEngineID`), the context name (`contextName`), the object identifier (OID – ex. `ifDescr`) and the instance identifier (ex. '`1`'). For SNMPv1 and SNMPv2c, queries require only the OID and the instance, which is more simple but less flexible.

The communication between SNMP entities follows a set of parameters, which identifies the communicating entities, security and the message processing. One of the fundamental

aspects of the communication process is the security parameters, which varies with the model version. For SNMPv1 and SNMPv2c, privacy is inexistent and the authentication is based on a community name. SNMPv3 already allows message encryption, which guarantees privacy and user based authentication.

For communication to succeed, it is necessary to indicate the protocol version. This is one of three choices: SNMPv1, SNMPv2c and SNMPv3. Moreover, it is necessary to associate with the protocol the destination address and the security parameters. These may be a single parameter for the first two versions – the community name string – or three parameters for SNMPv3:

- Security model – SNMPv1, SNMPv2c, USM (User Security Model).
- Username – character string.
- Security level – no authentication and no privacy (`noAuthNoPriv`), authentication and no privacy (`authNoPriv`), authentication and privacy (`authPriv`).

Moreover, it is necessary to pass to the security module the passwords or keys associated with the authentication and privacy protocols.

The communication protocol may also require two more parameters: a) the time to wait for an answer before failing (timeout) and b) the number of attempts before giving up (retry count).

Finally, to indicate the required resource it is necessary the context name, OID and instance.

C. SNMP URL

The SNMP URL collects all the information required for the communication between SNMP entities in a single character string following the URI format. According to the standard, a URI have restrictions in terms of symbols or characters not shown in this paper because of its extension. For further details, refer to [1].

Generically, the proposed syntax for the SNMP URL is:

```
snmpurl    = scheme "://" [security "@"] [host_port] ["/"
             [resource]["?"[operation]["?" [version] ["?"
             [context]]]]] ["#" parameters]
scheme     = "snmp"
security   = [community] |
             [user [":" auth [":" privacy]]]
community  = community - section 3.2.5 of [9]
user       = user name - section 2.1 of [10]
auth       = element_a ?("&" element_a)
element_a  = "auth=" protocol_a | "pass=" key
privacy    = element_p ?("&" element_p)
element_p  = "priv=" protocol_p | "pass=" key
protocol_a = authentication protocol- sect. 1.4.2 of [10]
protocol_p = privacy protocol - section 1.4.3 of [10]
key        = character string
host_port  = host [":" port]
host       = IP address or associated Internet name
port       = integer
resource   = OID ["/" instance]
OID        = '.' separated integers |     associated name
instance   = '.' separated integers
operation= "op=" op_name [op_params]
op_name    = name of the SNMP operation. For now:
             ("get"  |  "getNext"  |  "set"  |  "trap"  |
              "response"   |   "getBulk"   |   "inform"   |
              "trap2")
op_params  = op_param *("&" op_param)
op_param   = ("value="   value   |   "maxrep="   value   |
              "nonrep=" value)
value      = character string
version    = "v1" | "v2c" | "v3"
context    = context - section 3.3.1 of [11]
parameters =  parameter *("&" parameter)
parameter  = "timeout=" integer | "retries=" integer
```

Based on this syntax we can specify management information and services invocation in compact string as the following examples.

For SNMPv1:

snmp://private@sw1.estig.ipb.pt/sysContact/0?op=set&value=Rui

snmp://public@nms.estig.ipb.pt:161/sysUpTime?op=getNext

For SNMPv2c:

snmp://private@sw1.estig.ipb.pt/sysContact/0?op=set&value=Rui?v2c

snmp://public@nms.estig.ipb.pt:161/sysUpTime?op=getNext?v2c

For SNMPv3:

snmp://rlopes@sw1.estig.ipb.pt/sysContact/0?op=set&value=Rui?v3

snmp://guest@nms.estig.ipb.pt:161/sysUpTime?op=getNext?v3?router

The prefix '%' is used to represent special characters. The combination "%20" corresponds to the space character.

Note that although allowing keys and other security parameters in SNMP URLs, this procedure is not recommended because it relies on clear text. This inconvenient may be solved by direct interaction with the user (requesting specific security information on a separate window, for example) or, when there is no interaction with the user, storing the sensible information on separate encrypted files. This last choice is used in some SNMPv3 applications in the agent role [12].

III. Application Scenarios for SNMP URL

The first consequential shift related to SNMP URL usage happens at the API level. Usually, SNMP stacks require a large set of functions and a larger combination of function parameters. This varies with the version and with the product. For SNMPv3, for example, it is necessary to explicitly pass

the message processing model (associated with the protocol version), the security model, the security name, the security level, engine ID, context name, operation, and associated parameters. A single SNMP URL contains all this information.

As a consequence, it allows normalizing API function calling through different versions and different stacks. Moreover, it allows using different schemes to achieve higher-level protocol switching, such as HTTP or other.

To better understand and illustrate the previous concept we will discuss a set of practical usage scenarios. The first two show how they can be used on the agent side while the last shows an example for the manager side.

A. URL-TARGET-MIB

SNMP applications may contact other applications, possible remote, to retrieve some data or to send notifications. The SNMP architecture defines the SNMP-TARGET-MIB module to gather all the parameters necessary to the communication procedure and associates a tag list to them [13]. The application uses a tag to get the parameters from the SNMP-TARGET-MIB.

In other words, the SNMP-TARGET-MIB associates a list of names to a set of parameters that specify the host, port, protocol, security features and other information. For example, it may store the following information: to contact "router1" or "coreBridge" use the protocol=SNMPv3, timeout=5, retry count=3, user="senior", security model="USM" and security level="authNoPriv". These parameters are retrieved by the SNMP application by using a name, or tag, belonging to the previously referred tag list, in this case, "router1" or "coreBridge".

The other approach suggested in this paper is to associate a similar tag list to URLs in a single table, thus resolving individual tags to a single URI character string. This method, which we designate as URL-TARGET-MIB, allows complementing or even replacing the SNMP-TARGET MIB module (Fig. 1).

The main different between the modules is the extension. The use of URL-TARGET-MIB allows developing agents with less managed objects. Other advantage is to store OIDs and other URL schemes in the same MIB, which does not happen with the SNMP-TARGET-MIB.

B. Expression MIB modifications

The DISMAN charter defined under the IETF [14], a set of MIB modules to decentralize, by a set of distributed managers, some tasks traditionally associated with the central management station. From the agent point of view, these elements have the manager role and from the managers' point of view, these elements act as agents. In the latter, the management information deals with the configuration of management tasks. This model allows creating hierarchical management "islands" to increase the system robustness by introducing redundancy, scalability and by allowing operation in offline conditions.

One of the modules is the Expression MIB. It was defined to allow the definition of managed objects which where not considered during the definition of other modules [15]. It allows specifying expression based on existing managed

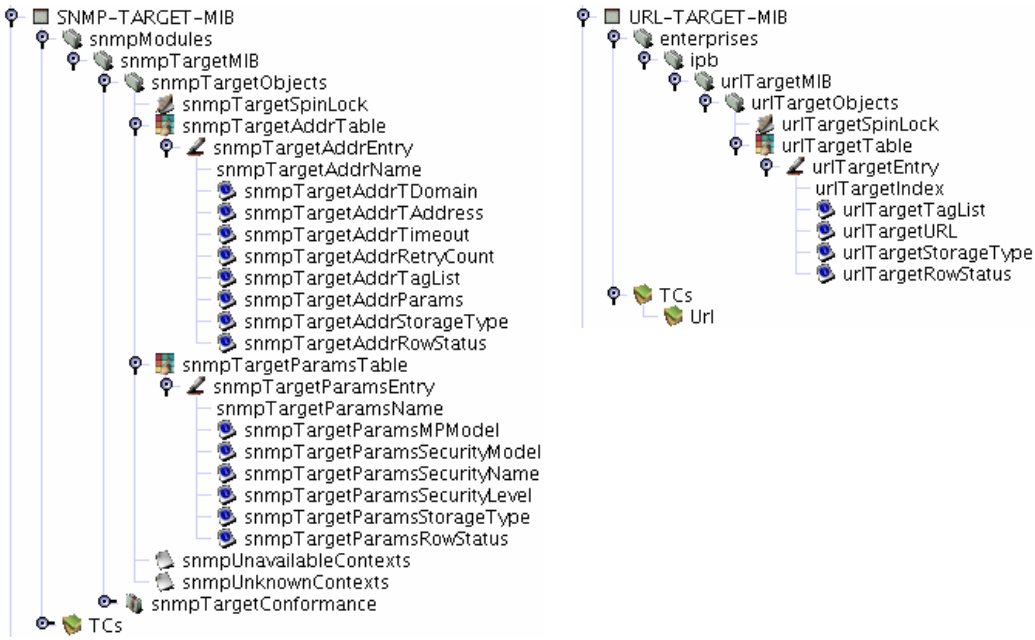

Fig. 1. SNMP-TARGET-MIB vs. URL-TARGET-MIB.

objects and it allows the construction of chaining expression, i.e., defining expressions which depend on other expressions results.

An expression is composed of operators, functions and values. The values may be constants or variables, the latter being associated with OIDs that refer to the correspondent value. A string defines each expression.

The possibility of using variables in expressions is, simultaneously, the strength and the weakness of the Expression MIB. As currently proposed, the MIB does not allow retrieving values from remote agents restricting the expression evaluation to local objects. This limits the possibility of creating some expressions, for example, when they use values from different sources.

The variables are defined in a separate table that contains a column with an OID, lacking other parameters necessary to communicate with remote agents, namely, the host address, port, version, security parameters and context. Any expression is defined according to the following generic format:

$$x = Expression(oid_1, oid_2, ... oid_n)$$

The usage of SNMP URLs instead of OIDs in the referred column allows giving the possibility to the Expression MIB to obtain values from remote agents thus increasing its flexibility and capability. It is than possible to use expressions like:

$$x = Expression(url_1, url_2, ... url_n)$$

This association may be performed in two ways. The first, just like in the Event MIB [16], adopts a specific MIB to store and index URLs. In this way, the expression variables are defined as URL references. The advantage of this approach is the compatibility with the SNMP-TARGET-MIB, defined to the SNMPv3 but requires an additional MIB module. The second, simpler approach, goes through replacing the OID column by a URL column, which allows replacing only the column data type.

C. *Management of Mobile Agents*

Mobile Agent technology is a paradigm mostly inherited from artificial intelligence "manuals". The novelty of this new paradigm is mainly associated with conception and abstraction more than the implementation technology. Basically, agents are software entities with the ability to perform actions on behalf of other programs, a person or an organization (the agent authority) [17]. A mobile agent is a specialization of these software agents that has the particular ability to migrate across several hosts. Mobile agents travel from node to node along the network carrying its state, so that they can maintain useful information when migrating to the next host. However, besides adding flexibility and the possibility to improve management efficiency it also brings some dilemmas, namely, it increases the agent management difficulty [18] and also introduces some kind of usability challenges and possibly threats [19].

This kind of programs or processes requires a platform, which provides the runtime environment and resources to individual agents. The diversity of currently available mobile agent platforms, coming from different vendors and supporting different languages, leads to the fact that mobile agents also need to be managed through a uniform way. The Mobile Agent Facility (MAF) specification is a CORBA based proposal from the OMG (Object Management Group), and it is a first attempt to standardize actions aiming at the interoperability between different vendors' agent systems [20]. Moreover, it provides yet some management facilities over mobile agents and allows a mobile agent to move between agent platforms with similar profile (sharing the language, authentication mechanism and serialization encoding). Fundamentally performed by proprietary tools, mobile agents' management may be thus performed in a standard, cross platform way.

In previous work, the authors have already proposed a MAF-MIB to convert between SNMP commands and MAF interfaces calls making possible to manage MAF compatible platforms in SNMP and MAF simultaneously [21].

In this context, we developed a prototype of a GUI tool for the management of mobile agents supporting two simultaneous access methods in a single interface: SNMP and CORBA (Fig. 2).

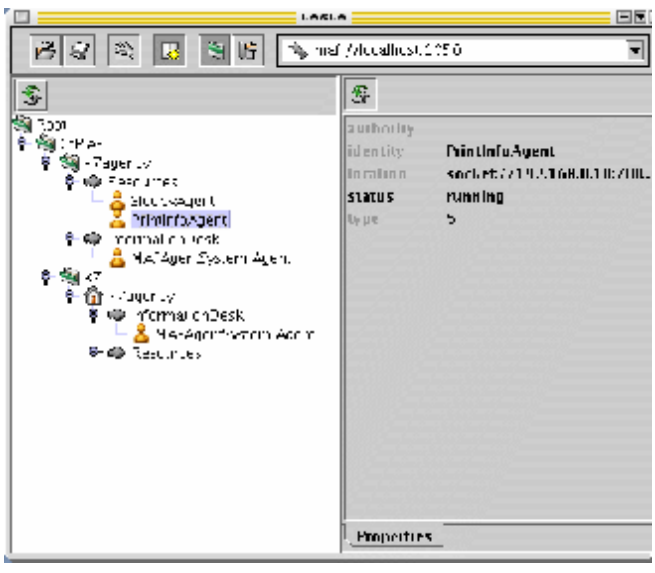

Fig. 2. URI based mobile agent management tool.

Just as for SNMP URLs, we defined a MAF URL with the format maf://<name service host address>:<port>/<context>. The common syntax, although semantically different, allows

defining a list of URLs (bookmarks or favorites) regardless of the management model. According to the URL scheme, the tool loads the appropriate module and proceeds along with the user commands.

The previous figure shows a tree view of the mobile agents' platform resources and was built after setting the URL in the address field (upper right corner). If the user sets an SNMP URL, the tree is modified according to the information from the SNMP agent.

IV. Future work

This concept, although simple, may have some impact on existing management tools. A guideline for future work is to study the impact that it may cause in existing MIB modules other than the Expression MIB. The Event MIB, Schedule MIB and RMON MIB, for example, look like perfect candidates to benefit from using SNMP URLs.

Another line of work, already in research by the authors, is the extension of the Expression MIB to allow URLs with different schemes. The possibilities for this approach are limitless because it would allow defining more functions as CGIs (or servlets or server side scripts), using different nature values and resources. The new function parameters may be passed by HTTP GET or POST operations and the value can be returned in MIME encoding. Moreover, it would allow defining expressions with values from different application fields, for example, relating the number of pages dispensed by an HTTP server with the available bandwidth.

In the Event MIB, it would than be possible to monitor resources directly, for example, LDAP, HTTP or FTP servers without needing intermediate SNMP agents. There is still a lot of work to be done, but the possibilities look immense.

V. Conclusions

The URI and URL concepts, although used frequently to identify and locate Internet resources do not have been used extensively in network management systems.

This paper suggests a URI scheme to identify SNMPv1, SNMPv2c and SNMPv3 resources. This approach, which we call SNMP URL, allows specifying in a single line of text all the parameters necessary for two SNMP entities to communicate, regardless of the version and the security model.

To validate the SNMP URL we also present some practical scenarios. The first example is an alternative approach to the SNMP-TARGET-MIB, using URLs to identify local or remote managed objects. Another example is an extension to the Expression MIB that allows evaluating expressions with any combination of objects from several agents. Just as for Internet browsers, it is also suggested the use of SNMP URLs in management stations to distinguish the service to use.

References

[1] T. Berners-Lee, R. Fielding, U.C. Irvine, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", Internet Request for Comments 2396, August 1998.

[2] J. Case, R. Mundy, D. Partain, B. Stewart, "Introduction to Version 3 of the Internet-standard Network Management Framework", *Internet Request for Comments 2570*, April 1999.

[3] R. Lopes, J. Oliveira, "SNMP Management of MASIF Platforms", *Proc. of the IFIP/IEEE International Symposium on Integrated Management – IM'2001*, Seattle, May 2001.

[4] iosnmp (http://www.opensnmp.org/).

[5] KDE (http://www.kde.org/).

[6] Cricket (http://cricket.sourceforge.net/).

[7] 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 – SSGRR'2000*, l'Aquila, Roma, Italia, August 2000.

[8] Uniform Resource Identifier (URI) SCHEMES (http://www.iana.org/assignments/uri-schemes).

[9] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)", *Internet Request for Comments 1157*, May 1990.

[10] U. Blumenthal, B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", *Internet Request for Comments 2574*, April 1999.

[11] D. Harrington, R. Presuhn, B. Wijnen, "An Architecture for Describing SNMP Management Frameworks", *Internet Request for Comments 2571*, April 1999.

[12] The NET-SNMP Project (http://www.net-snmp.org/).

[13] D. Levi, P. Meyer, B. Stewart, "SNMP Applications", *Internet Request for Comments 2573*, April 1999.

[14] DISMAN Charter (http://www.ietf.org/html.charters/disman-charter.html).

[15] R. Kavasseri, B. Stewart, "Distributed Management Expression MIB", *Internet Request for Comments 2982*, October 2000.

[16] R. Kavasseri, B. Stewart, "Event MIB", *Internet Request for Comments 2981*, October 2000.

[17] Pham V., Karmouch A., "Mobile Software Agents: An Overview", *IEEE Communications, Vol. 36, No. 7* (1998) pp. 26-37.

[18] M. Breugst, S. Choy, "Management of Mobile Agent Based Services", *Proc. of the 6th International Conference on Intelligence in Services and Networks, IS&N'99*, Barcelona, Spain, 1999.

[19] E. Kaasinen, "Usability Challenges in Agent Based Services", Proc. of the 6th International Conference on Intelligence in Services and Networks, IS&N'99, Barcelona, Spain, 1999.

[20] Mobile Agent Facility Specification, Object Management Group, 00-01-02.pdf (ftp://ftp.omg.org/pub/docs/formal/00-01-02.pdf).

[21] R. Lopes, J. Oliveira, "Descrição e Implementação de uma MIB para Sistemas MASIF", *Proc. of the 3rd Conference in Computer Networks – CRC2000*, Évora, Portugal, November 2000.