International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:6, 2008

Design of an SNMP Agent for OSGi Service Platforms

Pedro J. Muñoz Merino, Natividad Martínez Madrid, and Ralf E. D. Seepold

***Abstract*—**On one hand, SNMP (Simple Network Management Protocol) allows integrating different enterprise elements connected through Internet into a standardized remote management. On the other hand, as a consequence of the success of Intelligent Houses they can be connected through Internet now by means of a residential gateway according to a common standard called OSGi (Open Services Gateway initiative). Due to the specifics of OSGi Service Platforms and their dynamic nature, specific design criterions should be defined to implement SNMP Agents for OSGi in order to integrate them into the SNMP remote management. Based on the analysis of the relation between both standards (SNMP and OSGi), this paper shows how OSGi Service Platforms can be included into the SNMP management of a global enterprise, giving implementation details about an SNMP Agent solution and the definition of a new MIB (Management Information Base) for managing OSGi platforms that takes into account the specifics and dynamic nature of OSGi.

***Keywords*—** MIB, OSGi, Remote Management, SNMP.

I. Introduction

THE management of devices in a corporation was initially focused in network parameters and statistics. However, the increasing use of the Internet has connected new devices and applications, and as a consequence, the necessity of a dedicated management of applications or/and services has arised. For example, [1] explains a solution for including Siemens PLCs (Programmable Logic Controllers) features in remote management through SNMP (Simple Network Management Protocol).

Network parameters, applications or services can be managed through propietary protocols. Each device can set its own proprietary mechanisms for being managed through Internet or even through other proprietary networks. These solutions can be complete in terms of quantity and quality of managed parameters, but they are not interoperable. In this case, each device or type of devices needs to be managed independently from the others and the corporation cannot have a global vision of all their devices nor they can use standard protocols or tools.

The SNMP [2], [3] aims to provide a common protocol for management to allow interoperability. SNMP is de facto standard in network management and it is widely used in Internet. This is the reason why SNMP represents an advantage with respect to other management solutions such as those based on CORBA [4] (Common Object Request Broker Architecture) or RMI [5] (Remote Method Invocation).

On the other hand, as a consequence of the success of Intelligent Houses [6] they can be connected through Internet now by means of a residential gateway according to a common standard called OSGi [7] (Open Services Gateway initiative) that has been defined running in the residential gateway. The residential gateway is a platform where all the devices of an Intelligent House can be connected to. The residential gateway provides so called bundles that offer a set of services available for the devices connected to the platform.

These OSGi-based residential gateways can be included in the general schema of management as a new element, providing a remote control in terms of features, network parameters, services, applications, etc. This work provides a proposal of how to include residential gateways inside the global SNMP management framework. One of the main aspects is to define the features to be managed in SNMP thanks to a MIB (Management Information Base) [8]. OSGi-based residential gateways have a dynamical nature because services they provide can change dynamically (hot-plug). This implies to solve some problems in the definition of the MIB because MIBs store static information.

The remainder of this paper is organized as follows. The section "Related Work" describes related work in the area of remote management for OSGi Service Platforms. The section "Implementation details for an SNMP Agent for OSGi" describes some implementation decisions and the Agent functional block diagram. The second last section describes a new MIB proposal called OSGI-MIB for managing OSGi gateways that is according to the OSGi specific features and solves the dynamic nature problem. Finally, there is a section with our conclusions.

II. Related Work

There are commercial implementations of the OSGi gateways available that provide already SNMP solutions such as Prosyst [9]. The standard MIB that Prosyst provides is very limited. Nevertheless, it allows expanding the standard MIB

Manuscript received September 29, 2006. This work has been partly supported by the PlaNetS (Platforms for Networked Service Delivery) project (MEDEA+ Project A-121), financed by the Spanish Ministry of Industry (FIT -330220-2005-111).

Pedro J. Muñoz Merino is with the Department of Telematics Engineering, Universidad Carlos III de Madrid, Avda de la Universidad, 30 E-28911 Leganés (Madrid) Spain. Office: 4.1A08 (corresponding author phone: (+34) 91-624-8801; fax: (+34) 91-624-8749; e-mail: pedmume@it.uc3m.es).

Natividad Martínez Madrid is with the Department of Telematics Engineering, Universidad Carlos III de Madrid, Avda de la Universidad, 30 E-28911 Leganés (Madrid) Spain (e-mail: nati@it.uc3m.es).

Ralf E. D. Seepold is with the Department of Telematics Engineering, Universidad Carlos III de Madrid, Avda de la Universidad, 30 E-28911 Leganés (Madrid) Spain (e-mail: ralf@it.uc3m.es).

implementing new objects (scalars and tables) and even for new services. The Prosyst implementation provides a SNMP protocol and the user must map new parameters to new MIB objects. In addition, the user should write the JAVA code for the retrieval and setting of these new objects. Anyway, the new nodes of the MIB should be statically predefined and it does not allow a dynamic load of the new service data. Therefore, a manager should load the MIB each time there is a new object added. In this paper, we propose a MIB solution that is called OSGI-MIB. This OSGI-MIB should be loaded only once by a manager. If a new object is added, then the manager can find this new object due to the OSGi-MIB information, without the necessity of loading a new MIB. Furthermore, the OSGi-MIB proposed includes more information about OSGi than the Prosyst commercial solution.

There are other non-commercial OSGi solutions available at present that support remote management, like the one proposed with OSCAR using JMX [10]. JMX is a technology for getting and setting management information. JMX provides a JAVA oriented-base approach, which is indicated in dynamic environments such as OSGi platforms. Although JMX can be integrated with whichever management protocol such as RMI, CORBA or SNMP, however the commented solution does not explain the specific integration with SNMP. In order to use JMX together with SNMP, two adaptations should be required: 1) Adaptation of the retrieved information from JMX to a MIB. The description of dynamic information is easier in JMX than in SNMP due to the static nature of MIBs 2) Adaptation to translate from JMX to SNMP protocol.

III. Agent Implementation Details

Different OSGi gateway's aspects can be managed. Network administrators should decide which aspects are going to be managed. The proposed solution sets that the SNMP agent should be divided into sub-agents in which each one is in charge of a specific part. In a general case, it could be managed these different aspects: network parameters (for example with the MIB-2 [11]), services, the common OSGi framework and the OSGi common bundles. Fig. 1 shows this division in sub-agents.

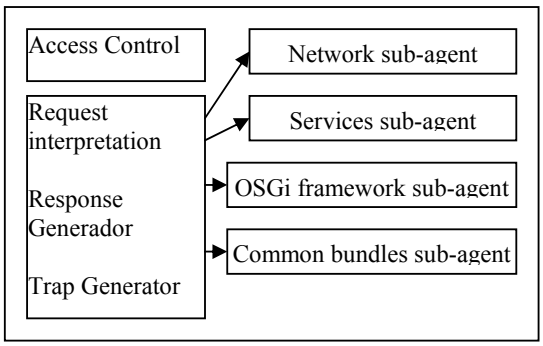

Fig. 1 Agent functional block diagram

Fig. 1 shows a possible implementation of an SNMP agent for OSGi gateways. This SNMP OSGi agent would be a bundle inside the OSGi gateway. Furthermore, this bundle should inherit from the OSGi management bundle, because it has permissions to access to some features that are needed to manage remotely. First of all, an Access Control Module checks if a specific manager has permissions in order to perform a specific action on the MIB. This is important in OSGi environments because there are different actors that can manage the OSGi platform and they will have different permissions. A possible scenario could be an operator that owns all management permissions and the user of the gateway with restricted permissions. But other scenarios with different profiles and permissions of managers can be defined.

In addition, there is a module where all the actions related to the RFC 1157 are supported. This module is in charge of receiving the manager's requests (get or set), interpreting the messages, parsing the ASN1 data into the required MIB objects and redirecting each object to the correspondent OSGi SNMP sub-agent. Next, when the sub-agents retrieve (get) or establish (set) the information for all objects, the module composes the response in order to send it to the manager. This module is also in charge of creating the trap messages that are sent as notifications to the managers.

Sub-agents are in charge of retrieving, setting and updating data from the MIB data base. Each sub-agent is dedicated to a specific task (network, services, common framework or common bundles). For example, the network sub-agent can retrieve and set information about network parameters and statistics (MIB-2 could be implemented here) but it cannot interpret the services of OSGi. The MIB (it will be presented in next section) is also divided into different branches for each task and each sub-agent is related to a specific MIB branch. Depending on the requested object OID (Object Identifier), the module in charge of the requests will redirect to the appropriate sub-agent. Each sub-agent get, updates and sets the specific information. In this way, it is allowed a common interface for management. Managers should not know about the internal sub-agents. In addition, each sub-agent can be modified independently, without modifying the rest of the agent implementation. This modular solution provides extensibility and scalability.

The mentioned scenario, it is valid for SNMPv1, but other SNMP version can be implemented based on it. SNMPv2 adds new primitives and heed manager's hierarchy, and SNMPv3 adds security. For OSGi gateways it is necessary to implement security aspects as well because it is critical that an attacker will not be able to control the OSGi gateway.

IV. MIB Definition Proposal for OSGi Gateways

Our new MIB creation includes relevant information for managing the OSGi framework, services and common bundles. The MIB has been created according to an analysis of OSGi (release 3) specification, selecting the parameters and information necessary for OSGi. The MIB does not include

information about network management. This information can be taken into account from other public MIBs as the MIB-2. Fig. 2 shows the general structure of our OSGi MIB.

The OSGi MIB has to model dynamic situations in two scenarios: 1) Bundles, registered services, etc. that are available in a moment in an OSGi gateway. This information changes during the time. 2) The information related to a service is not known a priori because whichever service could be added to an OSGi platform with whichever information.

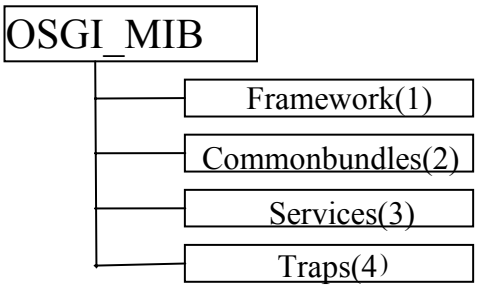

Fig. 2 General groups for the OSGi MIB

The OSGi MIB is divided into four groups:

• Framework: It provides information about the OSGi framework.

• Common bundles: This branch provides information about usual common bundles in OSGi.

• Services: This branch provides information about specific services in a moment. Each OSGi gateway can have different services in a concrete moment and new services can appear with unknown information.

• Traps: This branch contains a table defining the notifications that the manager configures with respect to other objects of the OSGI MIB.

A. Framework

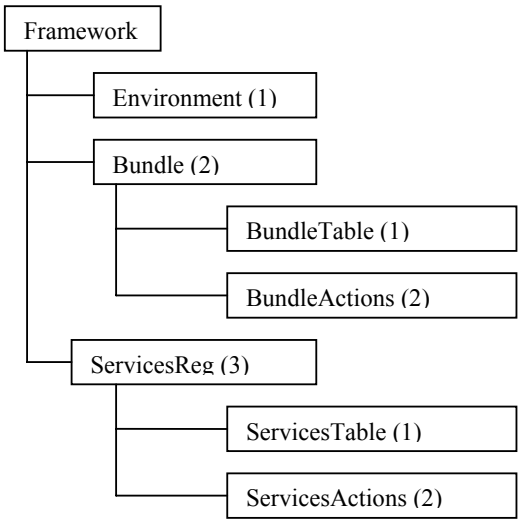

Fig. 3 Framework MIB branch

Fig. 3 shows the branches under the Framework node.

Under the Environment node, there are a set of scalar objects related to the OSGi framework environment. All are read-only (with the exception of startupFramework and stopFramework) so the information can not be modified by a manager. The environment parameters are: SPI (an integer that is the Service Platform Identifier), envversion (an integer that is the framework version), envvendor (an octect string that represents the vendor of the framework implementation), envlanguage (an octect string that represents the language to use in the framework), envexecution (an octect string that is a separated list of Execution Environments), envprocessor (an octect string that is the processor name), envosversion (an octect string that represents the version of the Operating System), envosname (an octect string that represents the name of the Operating System), startupFramework (this is a read-write parameter to start the framework), stopFramework (this is a read-write parameter to start the framework).

Table I shows the BundleTable branch.

TABLE I
BUNDLE TABLE BRANCH

OBJECT	OID	TYPE	ACCESS
BundleEntry	BundleTable.1	Aggregate	not-accesible
Bidentifier	**BundleEntry.1**	**Integer**	**read-only**
Bactivator	BundleEntry.2	OctectString	read-only
Bcategory	BundleEntry.3	OctectString	read-only
Bclasspath	BundleEntry.4	OctectString	read-only
Bcontactaddress	BundleEntry.5	OctectString	read-only
Bcopyright	BundleEntry.6	OctectString	read-only
Bdescription	BundleEntry.7	OctectString	read-only
BDocURL	BundleEntry.8	OctectString	read-only
Bname	BundleEntry.9	OctectString	read-only
Bnativecode	BundleEntry.10	OctectString	read-only
BreqEE	BundleEntry.11	OctectString	read-only
Bupdatelocation	BundleEntry.12	OctectString	read-only
Bvendor	BundleEntry.13	OctectString	read-only
Bversion	BundleEntry.14	OctectString	read-only
Bdynamicinput	BundleEntry.15	OctectString	read-only
BreqEEexport	BundleEntry.16	OctectString	read-only
BreqEImport	BundleEntry.17	OctectString	read-only
Bstate	BundleEntry.18	OctectString	read-only

The object in bold type is the index for the table. There are as many row instances as installed bundles (in a concrete moment). It is a dynamic table in the sense that the installed bundles number may change at any time. A manager can visualize the installed bundles that are available thanks to this table and all the information related to this bundle is available as well. The BundleTable columns objects are read-only and they represent information about a specific bundle: Bactivator (the class name for starting and stopping the bundle), Bcategory (all the category names), Bclasspath (all the JAR file names that should be searched), Bcontactaddress (the

contact address of the vendor of the bundle), Bcopyright (copyright specification of the bundle), Bdescription (description of the bundle), BDocURL (the URL where is located the documentation of the bundle), Bname (name of the bundle), Bnativecode (a specification of native code contained in this bundle), BreqEE (all the execution environments that are required for this bundle), Bupdatelocation (the location to retrieve the updated JAR file), Bvendor (bundle vendor), Bversion (bundle version), Bdynamicinput (all packages names to import dynamically), BreqEEexport (all packages names that can be exported), BreqEImport (all packages names that must be imported), Bstate that is the present state of the OSGi gateway (it can be resolved, started, stopped, or active).

The BundleActions branch contains the objects to install, desinstall, start, stop, and update the bundles of an OSGi gateway and to activate persistent storage for a bundle. There is one branch under BundleActions for each one of these actions. Fig. 4 shows the install branch under the BundleActions node.

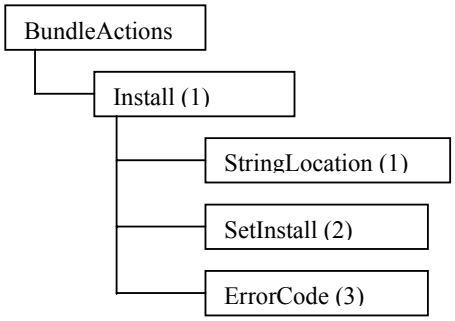

Fig. 4 Install bundle action branch

For example, for the Install Action, firstly the StringLocation object (which is read-write) should be written with the URL of this bundle location, next the SetInstall object (which is read-write) should be written with a '1'. By means of this, the bundle with location StringLocation will be installed in the OSGi gateway. Finally, the errorcode should be checked. The errorCode (read-only) will be a string which will contain the id of the last bundle that was tried to be installed and the error code in case of a not successful operation. When a successful install operation is done, then the BundleTable is modified because a new entry is added that corresponds to the new added bundle.

The desinstall operation is analogous to the install and when it is successfully applied an entry from the BundleTable is removed. Start and Stop operations are analogous to the install but the parameter that must be set before the operation is the bundle identifier instead of the location. Furthermore, a start or stop operation affects to the object state of an existing bundle inside the BundleTable. Moreover, the update operation updates the data of an entry of the BundleTable but does not create a new one. Finally, the Persistent Storage operation enables persistent storage for a bundle, but does not modify anything of the BundleTable.

On the other hand, the ServiceTable contains information about all the services that are in an OSGi gateway in a similar way that the BundleTable contains information about the bundles. For the ServiceTable, there are as many row instances as services. The column objects of the ServiceTable are all read-only and represent information about the services: Sidentifier (It is the identifier of the service), Sdescription (It is a short description of the service), Spid (This is the name registered service name), Sranking (This is the service ranking), Svendor (It is the service vendor), Sstate (It represents the service state in a moment. Its possible values are: modified, registered and unregistered), SOIDbundle (It is the OID of the owner bundle of this service in the BundleTable), SOIDMIB (It points to the OID of the root node where the sub-MIB for the service will be located. Under this OID will be all the specific information about this service.).

The ServicesActions branch has two sub-branches analogous to the BundlesActions, in order to register and unregister services. When a register or unregister operation is performed, then Sstate is changed in the ServicesTable.

B. CommonBundles

In this subsection it is analyzed which of the common bundles that are in the OSGiv3 specification can be managed. Under the Commonbundles branch, the bundles or packages that make sense to be managed are located as a subbranch for each one. These are the ones to be managed:

- Package Admin: It has two sub-branches in the MIB. The first sub-branch contains a table where all the packages for all the bundles of the OSGi gateway are shown. This table has a column called POIDBundle, where it is stored the OID of the bundle the package is exported from. The second sub-branch is related to the refresh action and it contains two scalar objects: BundleOID which is the OID of a bundle, and setrefresh which is the object to activate the refresh action of all the exported packages of the specific BundleOID bundle
- Start Level: This is to manage whichever start level for all the bundles and for the OSGi framework. This implies to add two new objects: StartLevel (a read-only integer) in the BundleTable that represents the start level for a specific bundle and SystemStartLevel (a read-write integer) in the EnvironmentTable that represents the start level for the entire system framework. Furthermore, under the StartLevel branch there are three scalar objects: 1)SLOIDbundle wich is a read-write parameter that indicates the OID of a bundle in the BundleTable. 2)SLstartlevelnumber which is a read-write integer that indicates the startlevel to be established in a bundle 3)SLsetstartlevel wich is a read-write parameter that when is set to '1' sets a new start level number (Slstartlevelnumber) in a bundle (SLOIDbundle).
- Permissions Admin: This is to establish permissions for all the bundles. It has two sub-branches: 1) A PermissionsTable which has as many entries as there are different established permissions. Each permission contains a

name, an action, the OID of the bundle which the permission references to and the type of permission (default or associated to a bundle) 2) PermissionsAction which contains all objects necessary to add, delete and update the permissions that are in the PermissionsTable.

• LogsTable: This is to retrieve information about the different logs in the system. This table has as many entries as logs have been registered. The columns of this table are read-only: Llevel (level of the log), LOIDbundle (the OID of the bundle which is associated to the log), Lexception (string that identifies the exception associated with an error in case there is an error), LOIDService (string that identifies the service associated with the log), Lmessage (the specific log message), Ltime (the time when the event associated to the log happened)

• Configuration Admin: This is to set configuration parameters for a specific service when it is registered. There are two different branches that correspond to two different configuration objects: 1)SingletonConfigurationTable that stores the single configurations for services. Each entry is a configuration property for a particular service. The index of the table is composed of CAOIDService (a read-write parameter that is the OID of the pid of the referred service) and CAorderofproperty (a read-only parameter that is the order of this property inside the properties configuration for a specific service). The properties of a specific service which defines its configuration is the union of all the entries of the table which have the CAOIDService of the specific service. Each entry of the table has a set of read-write column parameters related to the property: CAname (the name), CAdescription (the description), CAtype (the SNMP type), CAvalue (the value) 2) FactoryConfigurationsTable that stores the Factory Configurations for Service.

• User Admin: This is to add users, groups and associated groups to allowed actions. In our proposal there are three tables for defining respectively: users, groups and actions. There are also scalar objects for doing the creation, deletion and updating of users, groups and actions tables. Furthermore, there is a mapping table in the MIB that will relate users to groups. A user could be in several groups. The mapping will be done thanks to the OID columns related to users and groups. There is also a table to associate groups to actions. This table has the mapping between the OIDs of actions and OIDs of groups. There is a field of the table to indicate if the relationship is Basic or Required according to the OSGi specificationNumber equations consecutively with equation numbers in parentheses flush

C. Services

The previous ServicesTable provides information for viewing all the services that are in an OSGi gateway in a concrete moment. But there is not information about the management of specific parameters of a specific service. For example, a temperature sensor could be a service where additional information related to the temperature in a concrete moment in a house room may be required. This type of information is difficult to be static because OSGi platforms are not restricted to a predefined set of services and parameters to manage. Therefore, each service can have its own proprietary set of parameters to be managed and new services can be registered dynamically in OSGi platforms. Our MIB should take into account this behaviour. Fig. 5 shows services group when all elements of the commented service are scalars.

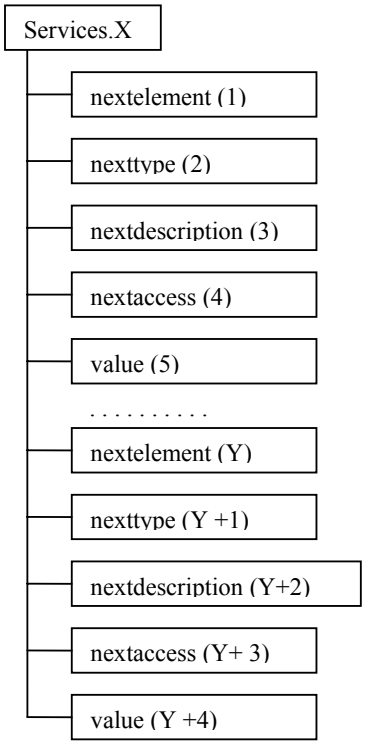

Fig. 5 Services group when all the service parameters are scalars

In a concrete moment, only services can be managed which are in the ServicesTable. Each of these services has a specific set of parameters to be managed. The sub-MIB in charge of taken into account the particular information about one service is referenced by the `SOIDMIB` column of the ServicesTable. The SOIDMIB (this is Services.X) indicates the root OID where all information is stored. It is not possible for two different services to have the same `SOIDMIB` (so two different services have a different X number). Fig. 5 shows the services group to take into account the dynamic service environment. There is a set of read-only objects that provides all the information about the next parameter of the service: nextelement (the name of the next parameter of the service. If no name is supplied then there is no more parameters for the service), nexttype (the SNMP type of the next parameter), nextdescription (the description of the next parameter), nextaccess (the access in the next parameter: read-only, write-only, read-write or not-accessible).

There are as many parameters associated to a service until the correspondent nextelement is '0'. It should be noted that with this MIB group, a specific service can change dynamically in the time, including or remaining some objects.

Furthermore, the number, names, etc. of parameters for a service are not predefined.

As a particular service example, consider that a temperature sensor service should be added to the OSGi platform with "temperature value", "vendor" and "precission_level" as parameters. Then the ServicesTable would have an entry that would represent this service. We would get the SOIDMIB for this entry, we will suppose it is .1.3.6.1.3.200.16 = Services.16, then the values in this services group would be:

Services.16.1="temperature value"
Services.16.2="INTEGER"
Services.16.3="Temperature in Celsius degrees"
Services.16.4="read-only"
Services.16.5="27"
Services.16.6="vendor"
Services.16.7="OCTECT STRING"
Services.16.8="temperature sensor vendor"
Services.16.9="read-only"
Services.16.10="ACZ"
Services.16.11="precission_level"
Services.16.12="INTEGER (0..4)"
Services.16.13="Upper number more precission"
Services.16.14="read-only"
Services.16.15="4"

D. Traps

The traps are related to two alarm SNMP tables that are under the trap node. The trap conditions can be configured thanks to these two tables. Traps are related to the different OSGi-MIB objects. A trap is sent from the agent to the manager when certain OSGi-MIB object is between certain value ranges. Whichever string or integer object of the OSGi-MIB (thus for example, possible values are the state of a bundle or the state of a service) can be used for configuring a trap. One table is dedicated to string objects and the other to integer ones. These two tables let set traps about MIB object conditions in a dynamic way.

V. Conclusion

This paper motivates the necessity of SNMP management for OSGi platforms thus allowing a scaleable remote management of complex devices in a dynamically changing environment. In order to achieve this objective, SNMP agent implementation details have been described.

A new OSGI-MIB has been designed that includes all relevant information to be managed in an OSGi platform. The MIB is based on relevant information described in the OSGI v3 specification. The dynamical nature of OSGi about bundles, packages, services, etc. requires a special MIB. Our OSGi-MIB proposal allows such dynamic management environment without assiging the SNMP manager role to change the initial MIB.

References

[1] P. J. Muñoz, "Diseño de un módulo de gestión de red para un PLC de Siemens S7", Master Thesis, Valencia, 2003. Award for the Best Master Thesis in Telematics Engineering for the Association of Spanish Telecommunication Engineering.
[2] Network Working Group, "A Simple Network Management Protocol (SNMP)", RFC 1157, may 1990.
[3] W. Stallings "SNMP, SNMPv2, SNMPv3, and RMON 1 and 2", 3rd Ed., Adisson-Wesley, 1999.
[4] S. Landis and S. Maffeis ''Building reliable distributed systems with CORBA,'' Theory and Practice of Object Systems, John Wiley & Sons Publishers, New York, 1997.
[5] V. Krishnaswamy, D. Walther, S. Bhola, "Efficient Implementations of of Java Remote Method Invocation (RMI)", Proceedings of the 4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS), 1998.
[6] Innovation Center Intelligent House, "INHAUS", http://www.inhaus-duisburg.de/en/index.htm
[7] OSGi Alliance, "OSGi Service Platform release 3", March 2003, available at http://osgi.org
[8] Network Working Group, "Structure and Identification of Management Information for TCP/IP-based Internets", RFC 1155, may 1990.
[9] Prosyst Software AG, "SNMP Package", Manual of Prosyst, 2001-2003, available at. http://www.prosyst.com/osgi.html
[10] M. L. Santillán, "Desarrollo de una herramienta de gestión remota de pasarelas de servicios domésticas", Master Thesis, Madrid, 2004.
[11] Network Working Group, "Management Information Base for Network Management of TCP/IP-based Internets: MIB-II", RFC 1213, march 199.