A CIM Framework for Standard-Based System Monitoring Using Nagios Plug-ins

Marcelo Lorenzati[1], Miriam Estela[1], Rodolfo Kohn[1]

[1] Intel Software Argentina, Software and Services Group,
Corrientes 122, Piso 2, Garden Center,
5000 Córdoba, Argentina
{marcelo.lorenzati, miriam.estela, rodolfo.kohn}@intel.com

Abstract. The Common Information Model is a widely-accepted industry standard to model distributed system objects as well as their behaviors and interactions to realize system management tasks. It is endorsed by the Distributed Management Task Force and appears as the preferred manageability solution to deal with the ever increasing heterogeneity characterizing today's datacenters. However, a number of enterprise-class system management products, like Nagios, are not compliant with this standard. Nagios is among the top open source monitoring tools with the power of a large community of developers producing plug-ins to manage a variety of enterprise systems. As part of the endeavor to accelerate CIM adoption, an extension framework, called Plugin Extension for CIM, has been developed in order to expose Nagios and other third-party plug-ins thru CIM, thus enhancing the capabilities of standard-based system management tools by the transparent use of the extensive variety of existing plug-ins. This paper describes the developed framework as well as its acceptance within the open source manageability community.

Keywords: Manageability Standards; Monitoring; CIM; Nagios; WBEM; Plug-ins.

1 Introduction

The ever increasing datacenter complexity seems to continue raising system management costs during the next years, dramatically incrementing their weight in the whole of IT expenditures. IT infrastructures need to support dynamic business processes relying on distributed data and distributed applications usually exposed as web services to the Internet and often subject to extremely variable and unpredictable workloads. These business processes are part of services that have to be delivered within the limits imposed by service level agreements and violations turn into monetary penalties to companies. While carrying out a number of tasks as change management, patching, system fault diagnosis and healing, or intrusion prevention and detection, IT professionals have to assure services scale out and are executed as expected without affecting availability and reliability [1]. Furthermore, a datacenter infrastructure includes tens or thousands of hardware and software resources of

different, and sometimes incompatible, technologies and vendors thus reducing or completely eliminating interoperability between devices, applications, and networks. This heterogeneity is one of the most significant factors adding to distributed systems management complexity.

In order to deal with the growing heterogeneity and achieve real integration of management information, the Distributed Management Task Force [2] embraced the Web-Based Enterprise Management (WBEM) [3] specification that includes the Common Information Model (CIM) [4] and various interoperable Internet technologies for distributed systems management [5]. CIM is an industry-accepted standard specification to model management information in distributed systems and to provide standard mechanisms, through CIM profiles, to realize a number of essential management tasks.

Notwithstanding, there are various renown system management products providing limited support of CIM if any, both in the open source and non-open source worlds. Among them, Nagios [6] is one of the most widely-used open source monitoring tools in the IT industry. Its power mainly lies on both its plug-in-based architecture and the huge community of developers devoted to the creation of a vast number of plug-ins providing a broad range of management capabilities. However, it relies on a proprietary approach to expose these manageability plug-ins to the core monitoring engine.

As an effort to contribute to the adoption of CIM and enhance the potential of standard-based monitoring tools, we developed a CIM extension framework that allows a CIM broker to expose Nagios, and other third-party, plug-ins through a CIM interface. Thus standard-based monitoring tools can transparently benefit from thousands of new manageability features in a matter of seconds by configuring and deploying already existing plug-ins that instrument the system and generate alarms whenever certain measured values are outside the specified ranges, usually because of an error state.

The framework consists of a CIM extension schema, a configuration script, and a generic plug-in indication provider. The script is run at the beginning to configure the parameters used to execute each plug-in that is to be run in the monitored system. The indication provider is the software that executes the requested plug-in with the appropriate parameters whenever a CIM client subscribes a listener to receive the corresponding indications. The provider parses a filter embedded in the indication subscription, based on this filter reads the relevant plug-ins and their configuration parameters, and executes each plug-in with certain frequency defined among the parameters. As in the case of Nagios, a plug-in can be written in any language including scripting languages. When a plug-in detects a problem, the indication provider converts the output describing a warning or critical alarm into a CIM indication that is routed to the listener specified in the subscription. The proposed extension schema defines a set of classes necessary to carry out the mentioned activities.

The complete framework is called “Plugin Extension for CIM”. It is currently offered as a downloadable manageability package, called sblim-cmpi-pec, belonging to the open source project Standards Based Linux Instrumentation (SBLIM) [12]. This is an umbrella project run by IBM that encompasses a collection of system

management tools to enable the implementation of WBEM in Linux. SBLIM packages are shipped within various Linux distributions such as SuSE.

Other efforts have been carried out in the industry to integrate the best of existing system management tools with WBEM standards. As an example, it is possible to mention the work done by Novell Inc. to expose statistical data obtained by Ganglia Monitoring System about data center systems thru the CIM statistical model. As of today, this seems to be the only project that integrates Nagios plug-ins to be exposed thru CIM and intends to enable Nagios engine to use CIM.

This paper describes the whole framework developed, its advantages for datacenter management, its contribution to industry standards adoption, and future work that includes more features and follows the converse way permitting Nagios core engine the transparent use of CIM-based functionalities. Section II provides a brief introduction to the main technologies involved, section III describes the framework implementation, section IV explains further work necessary, and finally the conclusions are stated emphasizing the contributions to the industry.

2 Technology description

2.1 The Common Information Model (CIM)

The Common Information Model is a standard to represent distributed system management information and define the objects and interactions required to realize specific system management tasks.

CIM is an object oriented model described with UML that includes classes, properties, methods, indications, associations and allowing subclassing for extension, but enforcing basic object hierarchy, abstraction and encapsulation. The syntax language to define the elements of CIM in a text format is Managed Object Format (MOF) which is based on Interface Definition Language (IDL).

This model provides a common definition of management information for systems, networks, applications and services, and allows for vendor extensions. The CIM's common definitions enable vendors to exchange semantically rich management information between systems throughout the network.

The CIM is made of two parts. The first one is the CIM Specification, which defines the language (syntax and rules) and the proper methodology for describing the management data. It also defines the CIM meta schema, its elements and the rules of each one of its elements.

The second part is the CIM Schema, which provides the classes and associations and its attributes as a complete Framework. It is comprised by the Core Model, Common Models and the extended Schema.

The Core Model defines all the concepts that are applicable to all the areas of management, and the Common Models are various models that define concepts that although they are common between a certain area of management, they are still independent of the implementation, technology or manufacturer.

Finally, the Extension schema expresses the technology specific concepts represented by classes, methods, attributes and associations, extending the common and core models.

Figure 1 shows part of the Core Model and Systems Model and two classes extending the schema.

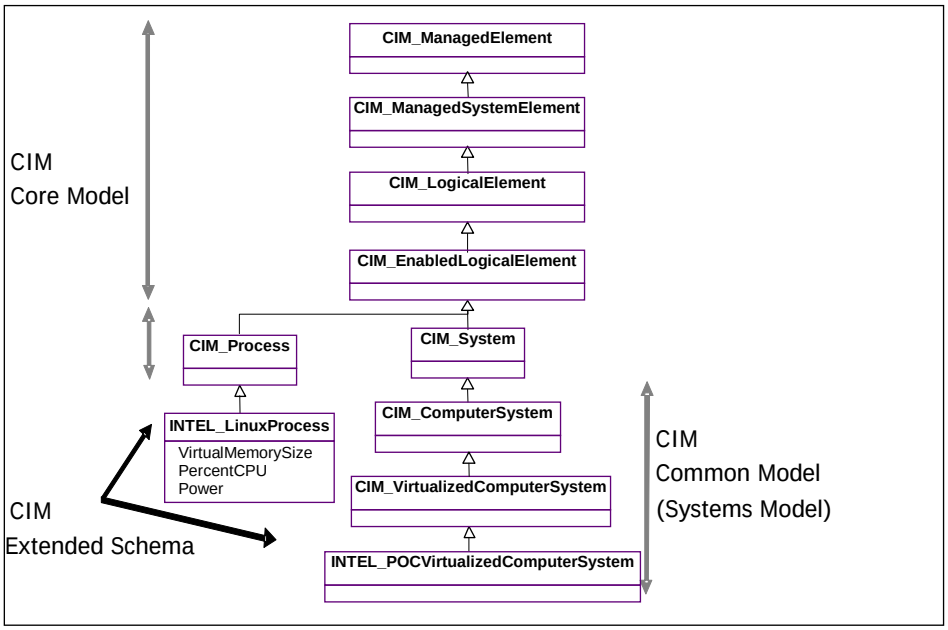

Fig. 1. A view of the Core Model and System Model hierarchies with two classes extending the CIM schema.

Figure 2 shows a CIM server consisting of a CIM Object Manager (CIMOM) or CIM Broker that interfaces with a client through a CIM-XML adapter and an indication handler. It receives requests from a client and has an interface with the providers to perform the required actions. There are various CIM broker implementations like SFCB [7], openWBEM[9], OpenPegasus[10], and Microsoft's WMI [11].

CIM Providers instrument the system for manageability. There are different types of providers: Instance, Association, Property, Method and Indication Provider. The CIMOM invokes providers in order to realize management tasks on the CIM models.

The Common Manageability Programming Interface (CMPI) is a standard interface used between the CIM broker and a provider, so that a CMPI-compliant provider is portable through different CIM brokers.

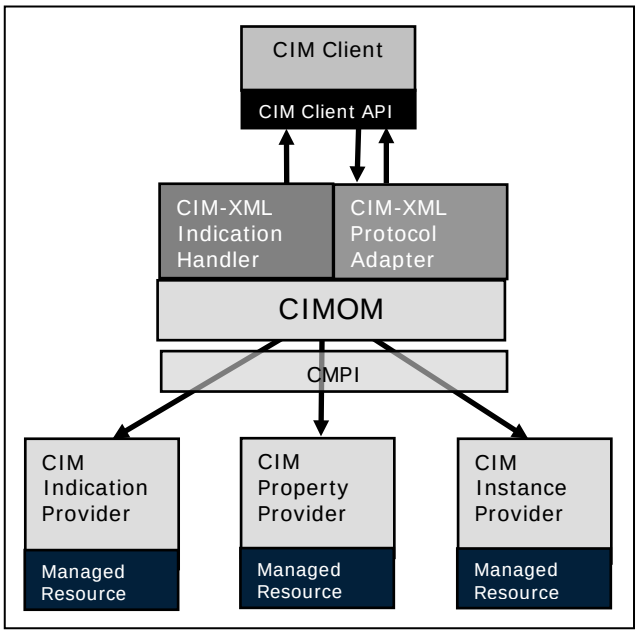

Fig. 2. A CIM Server implementation.

CIM defines the concept of indications in order to report changes in the state of the environment. For any event in the system, a CIM indication may be generated by the corresponding CIM Indication provider developed to monitor a specific aspect of the system.

When a client needs to receive notifications for specific types of event, it has to send a subscription for the corresponding indications specifying the location of a listener that is to receive any indication generated from the mentioned subscription. This subscription may also include a filter in order to refine the scope of the events that need to be notified.

There are 3 main types of indications: CIM_InstIndication, CIM_ClassIndication, CIM_ProcessIndication.

The indication provider developed to execute Nagios plug-ins generates a sub-type of process indications called alert indications.

2.3 Nagios

Nagios is a widely used open-source monitoring tool for host systems and network services. It is built following a server/agents architecture. Usually, on a network, a Nagios server is running on a local host, and plug-ins are running on all the remote hosts that need to be monitored. These plug-ins send information to the server, which

displays it in through a Graphical User Interface (GUI). It allows administrators to manage complex infrastructures by monitoring the overall status trough a web browser interface and subscribe to alerts notifications trough email, instant message and SMS when problems happens.

Two basic monitoring entities are defined by Nagios: the host that is the physical or virtual asset to be monitored; and the services that are the real monitored instances or processes running on a given host. Nagios call-back mechanism is one-way, informational-only.

The Core building block is formed by 3 parts:

- The scheduler is the server part of Nagios. At regular interval, the scheduler checks the plug-ins, and according to their results, it performs some actions.
- The Nagios GUI: it is displayed in web pages..
- Plug-ins: small programs (in Perl, C, java, python among others) that check a service and return a result to the Nagios server.

When a plug-in is executed, it generates data sent to the core engine which processes them and runs event-handlers notifications depending on the case. The plug-in returns a value and a small line of text Plug-in output according to Nagios specifications. The plug-in must return one of a set of possible values (ok, warning, critical) and it must print information text to the standard output. Because the plug-ins are configurable by the user, their output depends on the parameters, options and the check that they carry out.

3 Exposing Nagios plug-ins through CIM

3.1 Framework Description

Plugin Extension for CIM (PEC) consists of a CIM extension schema, a configuration script, and a generic plug-in indication provider.

Each Nagios plug-in is executed passing specific options and arguments to obtain certain functionality. The options and arguments are edited in a configuration file to be read by the configuration script which populates the PEC schema in the CIM database. This configuration script uses the CIM client library SFCC [8].

The PEC indication provider is the software that, when the appropriate CIM subscription arrives, executes the requested plug-ins with the appropriate options and parameters read from the database, and sends the corresponding indication depending on the plug-in output.

The provider executes each plug-in in a different thread. A thread executes the plug-in as a forked child process, at a configured frequency. The plug-in's output is read through an unnamed pipe. As a new process is forked to run a plug-in, plug-ins can be developed in any programming or scripting language.

The provider converts plug-in WARNING and CRITICAL output into CIM indications, with the appropriate severity levels, that are routed to the listener specified in the subscription.

The CIM extension schema defines a set of classes and associations necessary to carry out the mentioned tasks.

3.2 Extended PEC schema

In the proposed solution, the extended model consists mainly of four classes as shown in Fig. 3.

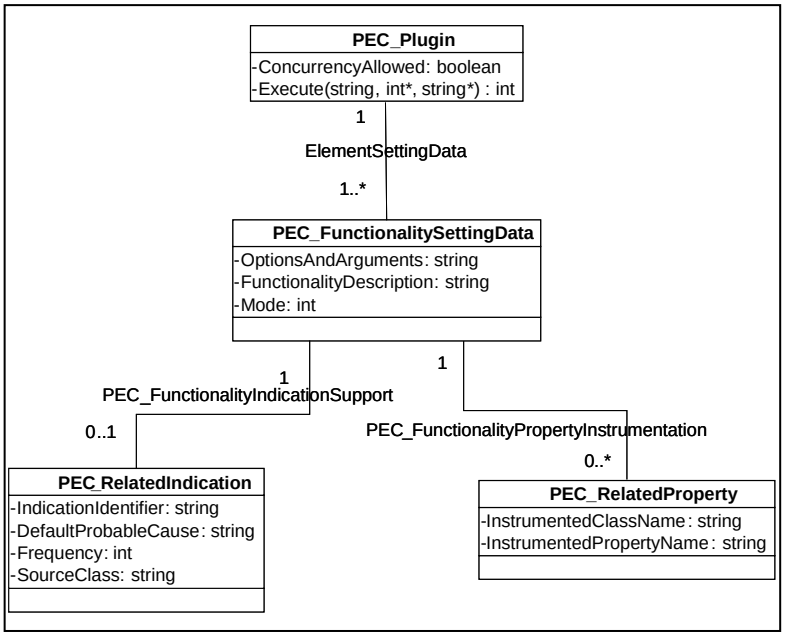

Fig. 3. Extended PEC model.

The class PEC_Plugin represents the plug-in in the system, where its main method is Execute. Each plug-in could have one or more functionalities depending on the options and parameters configured in PEC_FunctionalitySettingData. The plug-in can be used to monitor the system through the PEC indication provider or to obtain a specific property value for another object. In the first case the functionality is associated to an indication defined in PEC_RelatedIndication. In the second case, the functionality is associated to a class property specified in PEC_RelatedProperty. The associations between these classes are represented by ElementSettingData, PEC_RelatedIndication, and PEC_FunctionalityPropertyInstrumentation. The usage of plug-ins to monitor the system and generate indications is the one implemented at

the moment of writing this paper. The functionality to populate properties has not been implemented yet.

Figure 4 shows how these classes are connected to the CIM core and common models. PEC_Plugin actually inherits from PLUGIN_Executable which in turn inherits from CIM_SoftwareElement in the CIM Application model. PEC_FunctionalitySettingData inherits from CIM_SettingData in the core model and the other classes inherit from CIM_ManagedElement in the core model. Both figures 3 and 4 depict the attributes and methods of each class and association.

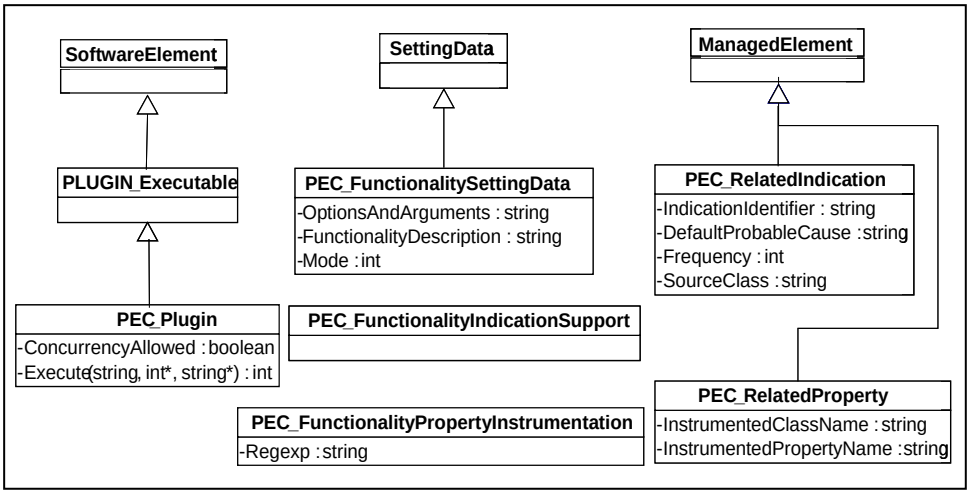

Fig. 4. Class hierarchy showing how the schema extends by inheritance the core and common models.

3.3 CIM-based Monitoring with Nagios plug-ins

The indication provider is registered to the CIM Object Manager through the compilation and installation of the PEC project with the GNU autotools (.configure make and make install). For our tests, the CIM Object Manager SFCB (Small Footprint CIM Broker) was used but since our plug-in has been developed according to the Common Manageability Programming Interface specifications, it is portable through any other CIM Object Manager.

The Nagios plug-ins to execute have to be deployed in the monitored system, usually storing them in a specific directory. The provider will require configuring the path to the Nagios plug-in by setting an environmental variable (PEC_PLUGIN_PATH) in order to execute the selected plug-ins.

The CIM repository has to be populated as explained above through the configuration file and scripts. The scripts connect to the CIM broker through the library SFCC.

Once the repository has been populated the indication provider can be executed when the appropriate subscription is received by SFCB.

In order to subscribe to indications, a client reference tool has been developed. This client subscribes for indication passing a specific filter. These filters can include logical operators as AND, OR and NOT, also allows to apply arithmetic operators like '>', '<' or '='.

The filter can only include the properties IndicationIdentifier and SourceClass that can be found in the class CIM_AlertIndication and PEC_RelatedIndication.

When a subscription is received the indication provider gets the appropriate plug-ins according to the filter received in the subscription. Then it reads the configuration data for each plug-in and executes it with the obtained options and arguments to realize the corresponding functionality.

Whenever the plug-in is run it can generate a WARNING or CRITICAL output depending on the monitored system state and the options passed for execution. In this case, the indication provider sends the configured indication number to the subscribed listener.

4 Future Work

At this moment the components to monitor a system, using Nagios plug-ins with CIM-based tools, have been developed. The package can be downloaded from SBLIM web site. Further work is necessary to implement the components to populate CIM class properties using third-party plug-ins. This would involve creating the appropriate libraries that would need to be linked into the corresponding CIM class providers which are to call the corresponding function wrapping a plug-in whenever the appropriate request is received.

Additionally, this work should be extended to enable a Nagios engine to use CIM-XML or other WBEM protocol, such as WS-Management, to access CIM elements. This work would have advantages like the capacity to take CIM-based actions as a response to an alarm, to seamlessly reuse CIM providers for monitoring, and using standard web-service interfaces that would have benefits such easier network boundaries traversal.

5 Conclusions

This paper describes a work that permits extending the capabilities of system management tools that are compliant with WBEM industry standards defined by the Distributed Management Task Force. Using the "Plugin Extension for CIM" package, a standard-based tool can transparently use available Nagios plug-ins and other third-party plug-ins, developed by a huge open-source community, to obtain thousands of new monitoring functionalities with no much effort. Thus, these system management applications can incorporate the monitoring of a large number of server aspects, including hardware and software, in a matter of seconds by just configuring the plug-in usage in a configuration file. After the configuration phase is concluded, these

plug-ins can be seamlessly used and indications can be generated when the measured system state is not within the configured range.

As a result of this work, it is easy to combine the benefits of the huge diversity of monitoring capabilities provided by Nagios plug-ins with the advantages of using a standard interface like CIM that allows overcoming the complexity generated by heterogeneous distributed systems.

The "Plugin Extension for CIM" framework has been donated to the open source project SBLIM, run by IBM. It can be downloaded from Sourceforge as a package called sblim-cmpi-pec [12].

6 References

1. J.O. Kephart and D. Chess, "The Vision of Autonomic Computing," Computer, vol. 36, no. 1, 2003, pp. 41-50
2. DMTF: www.dmtf.org
3. DMTF. Web-Based Enterprise Management (WBEM) standards, Retrieved January 12, 2009 from http://www.dmtf.org/standards/wbem
4. DMTF. Common Information Model (CIM), Retrieved January 12, 2009 from http://www.dmtf.org/standards/cim
5. A. Westerinen and W. Bumpus, "The Continuing Evolution of Distributed Systems Management," IEICI Transaction on Information and Systems, vol. E86-D, no. 11, 2003/11/01, pp. 2256-2261
6. Nagios project home page, Retrieved May 5, 2009 from http://www.nagios.org
7. SBLIM SFCB project home page, Retrieved May 5, 2009 from http://sblim.wiki.sourceforge.net/Sfcb
8. SBLIM SFCC project home page, Retrieved May 5, 2009 from http://sblim.wiki.sourceforge.net/Sfcc
9. OpenWBEM project home page, Retrieved May 5, 2009 from http://openwbem.sourceforge.net/
10. OpenPegasus project home page, Retrieved May 5, 2009 from http://www.openpegasus.org
11. Microsoft home page, Retrieved May 5, 2009 from http://www.microsoft.com
12. SBLIM project home page, Retreived May 5, 2009 from http://sblim.wiki.sourceforge.net