Standards Supporting Autonomic Computing : CIM Remodeling “LocalizationCapabilities” class in Core Model

A Thesis
Presented to the Graduate School,
Faculty of Engineering – Alexandria University - Egypt

For the Degree of
Master of Science
In
Computer Science

Field of Specialization
Autonomic Computing

By
Rania Al-Maghraby

2007

TABLE OF CONTENTS

List of Figures

List of Tables

List of Listings

List of Abbreviations

CIM.	Common Information Model
DMTF.	Distributed Management Task Force
ICU.	International Components for Unicode
CLDR.	Common Locale Data Repository
MDA.	Model Driven Architecture
IT.	Information Technology
MOF.	Managed Object Format
UML.	Unified Modeling Language

Acknowledgement

I'm thankful to every one helped me in preparing this material, specially mentioning :

Dr. Mohammed Salah Selim – Faculty of Engineering, Computer Science Dept., Alexandria University, for his effort and patience in supervision.

Dr. Nagwa M. El-Makky – Faculty of Engineering, Computer Science Dept., Alexandria University, for her cooperation and directions.

Aaron E. Merkin – CIM development team in IBM – Member of DMTF organization, for his significant comments and review of proposed model.

Standards Supporting Autonomic Computing : CIM Remodeling “LocalizationCapabilities” class in Core Model

Abstract

Autonomic Computing is a steadily emerging and promising research field. It aims at simplifying interoperability to diminish the management complexity in several industries. To this goal, some standards have been worked on to establish a common basis for communication and interaction on all management areas. This work focuses on one of these standards, CIM – developed by DMTF, trying to enhance its support for localization handling by proposing some modifications and additions that are intended and designed to facilitate the configuration of different localization aspects.

1. Introduction

As enterprises strive to meet their current challenges, they require an IT infrastructure that supports their business goals. An IT infrastructure that enables business to be more responsive, variable, focused, and resilient.

Autonomic systems are such systems that are self-configuring, self-healing, self-protecting and self-optimizing. They are “Intelligent” open systems that manage complexity, know themselves, continuously tune themselves, adapt to unpredictable conditions, prevent and recover from failures and provide a safe environment. They let enterprises focus on business, not on IT infrastructure.

For autonomic computing technology, many relevant standards and their associated standards organizations are described in "An Architectural Blueprint for Autonomic Computing" (see References[2]). In our work, we examine selected autonomic standard in detail; CIM.

More specifically, the handling of CIM model is conducted with concentration on the localization support features in the current version of the model (V2.15), by enhancement and additions to it to be more mature in dealing with localization configuration settings.

Software localization is, by global consensus, an extremely important issue in IT industry. It’s agreed upon that understanding different cultures and marketing for them is a critical success factor. Generally, Localization means allowing an application/product interface to adapt with specific locales (languages, countries, cultures, …). There are several techniques and tools that support localization.

The LocalizationCapabilities class in CIM core model was introduced in previous versions of CIM (V2.9) with the purpose of supporting diagnostic service data, with the intent that it will be generalized for broader use in future. It was then deprecated upon a change request to replace localization features in the schema by usage of the protocol CIM/XML over HTTP. Refer to more details on criticism of current design and rationale and significance of proposed work in section 6.4.

2. Autonomic Computing Overview

2.1. Definition

The term “autonomic” comes from an analogy to the autonomic central nervous system in the human body, which adjusts to many situations automatically without any external help. We walk up a flight of stairs and our heart rate increases. If it is hot, we perspire. If it is cold, we shiver. We do not tell ourselves to do these things, they just happen. Similarly, the way to handle the problem of managing a complex IT infrastructure is to create computer systems and software that can respond to changes in the IT (and ultimately, the business) environment, so the systems can adapt, heal and protect themselves. [5]

The IBM Autonomic Computing Initiative (first proposed in 2001) is an industry-leading effort focused on managing complexity. “Autonomic Computing” is IBM’s term for an approach and blueprint including a set of products, tools and services that add self-managing capabilities to Information Technology systems. Its goal is to shift the burden of support tasks such as configuration, maintenance, and fault management from people to technology.

Autonomic computing systems consist of four attributes. As illustrated in the following 4-quadrant chart, they are:

_ Self-configuring (able to adapt to changes in the system)
_ Self-healing (able to recover from detected errors)
_ Self-optimizing (able to improve use of resources)
_ Self-protecting (able to anticipate and cure intrusions)

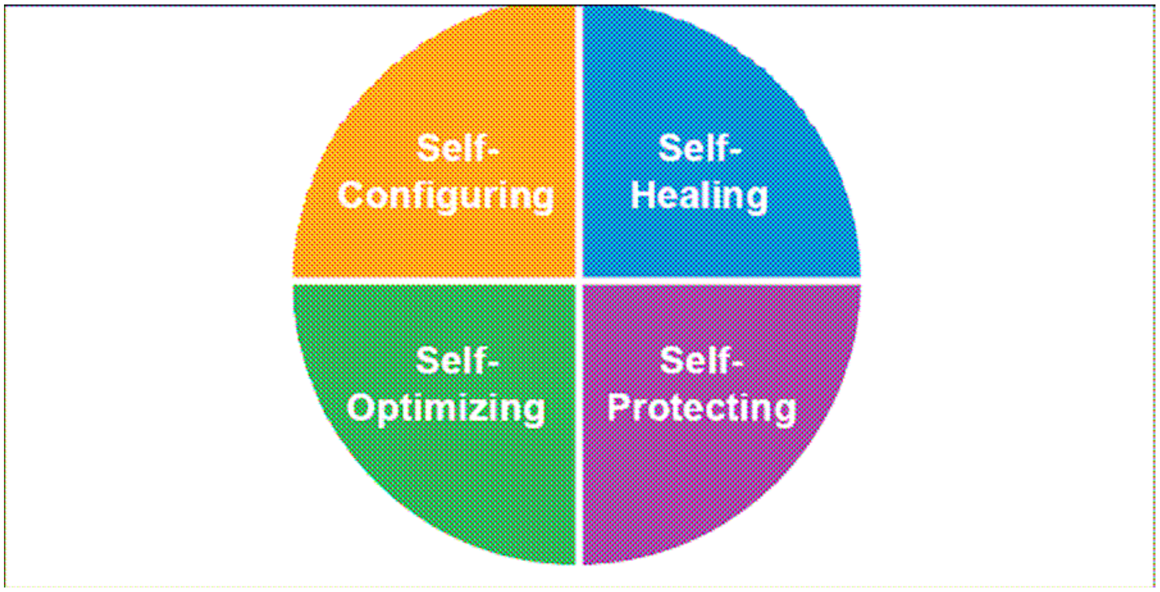


Figure 1. Autonomic Computing Characteristics

2.2. Autonomic Computing Concepts

In an autonomic environment, components work together, communicating with each other and with high-level management tools. They can manage or control themselves and each other. Components can manage themselves to some extent, but from an overall system standpoint, some decisions need to be made by higher level components that can make the appropriate trade-offs based on policies that are in place. The following figure represents the control loop that is the core of the autonomic architecture.

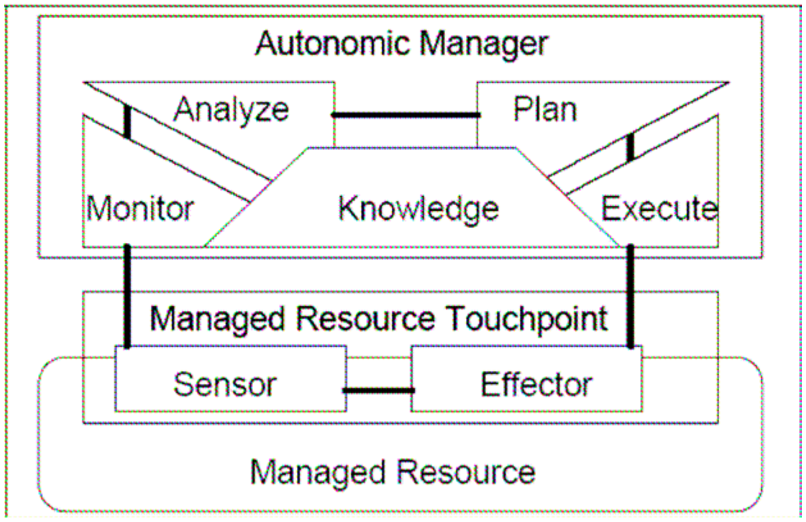


Figure 2. Autonomic Computing Control Loop

The autonomic manager implements autonomic control loops by dividing them into four parts: monitor, analyze, plan, and execute. The control loop carries out tasks as efficiently as possible based on high-level policies.

- Monitor: Through information received from sensors, the resource monitors the environment for specific, predefined conditions. These conditions don't have to be errors; they can be a certain load level or type of request.
- Analyze: Once the condition is detected, what does it mean? The resource must analyze the information to determine whether action should be taken.
- Plan: If action must be taken, what action? The resource might simply notify the administrator, or it might take more extensive action, such as provisioning another hard drive.
- Execute: It is this part of the control loop that sends the instruction to the effector, which actually affects or carries out the planned actions.

The managed resources are controlled system components that can range from single resources such as a server, database server, or router to collections of resources like server pools, clusters, or business applications.

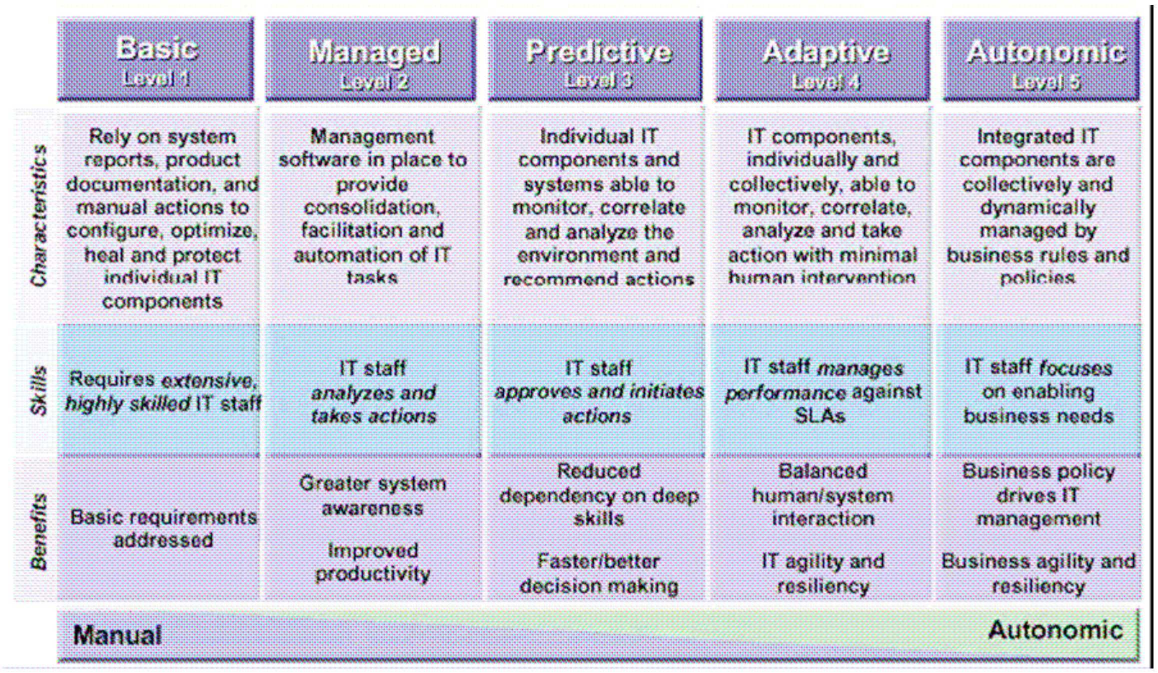
All of the actions in the autonomic control loop either make use of or supplement the knowledge base for the resource. For example, the knowledge base helps the analysis phase of the control loop to understand the information it's getting from the monitor phase. It also provides the plan phase with information that helps it select the action to be performed.

2.3. Autonomic Maturity Levels

It would be unreasonable to expect all software and system administration processes to suddenly go from a completely manual state to a completely autonomic state. Fortunately, with the autonomic computing model, that's not a requirement. In fact, there are typically five levels of autonomic maturity, which are measures of where any given process is on that spectrum. These maturity levels are:

- Basic -- Personnel hold all of the important information about the product and environment. Any action, including routine maintenance, must be planned and executed by humans.
- Managed -- Scripting and logging tools automate routine execution and reporting. Individual specialists review information gathered by the tools to make plans and decisions.
- Predictive -- As preset thresholds are tripped, the system raises early warning flags and recommends appropriate actions from the knowledge base. The centralized storage of common occurrences and experience also leverages the resolution of events.
- Adaptive -- Building on the predictive capabilities, the adaptive system is empowered to take action based on the situation.
- Autonomic -- Policy drives system activities, including allocation of resources within a prioritization framework and acquisition of prerequisite dependencies from outside sources.

Most systems today are at the basic or managed level, though there are, of course, exceptions.



	Basic Level 1	Managed Level 2	Predictive Level 3	Adaptive Level 4	Autonomic Level 5
Characteristics	Rely on system reports, product documentation, and manual actions to configure, optimize, heal and protect individual IT components	Management software in place to provide consolidation, facilitation and automation of IT tasks	Individual IT components and systems able to monitor, correlate and analyze the environment and recommend actions	IT components, individually and collectively, able to monitor, correlate, analyze and take action with minimal human intervention	Integrated IT components are collectively and dynamically managed by business rules and policies
Skills	Requires *extensive, highly skilled* IT staff	IT staff *analyzes and takes actions*	IT staff *approves and initiates actions*	IT staff *manages performance* against SLAs	IT staff *focuses* on enabling business needs
Benefits	Basic requirements addressed	Greater system awareness Improved productivity	Reduced dependency on deep skills Faster/better decision making	Balanced human/system interaction IT agility and resiliency	Business policy drives IT management Business agility and resiliency
	Manual				Autonomic

Figure 3. Autonomic Maturity Levels

2.4. Value of Autonomic Computing

By enabling computer systems to have self-configuring, self-healing, self-optimizing and self-protecting features, autonomic computing is expected to have many benefits for business systems, such as reduced operating costs, lower failure rates, more security, and the ability to have systems that can respond more quickly to the needs of the business within the market in which they operate.

The implications for an autonomic, on demand business approach are immediately evident: A network of organized, smart computing components can give clients what they need, when they need it, without a conscious mental or even physical effort.

Few examples of the results delivered by implementing autonomic computing solutions with self-management characteristics are : Operational efficiency, Supporting business needs with IT, Workforce productivity. Systems that are self-managing free up IT resources which then can move from mundane system management tasks to focusing on working with users to solve business problems.

2.5. Architectural Overview

In order for the autonomic managers and the managed resources in an autonomic system to work together, the developers of these components need a common set of capabilities. This section conceptually describes an initial set of core capabilities that are needed to build autonomic systems. These core capabilities include:

_ <u>Solution installation : </u>A common solution knowledge capability eliminates the complexity introduced by many formats and many installation tools. By capturing install and configuration information in a consistent manner, autonomic managers can share the facilities as well as information regarding the installed environment.

_ <u>Common systems administration :</u> Autonomic systems require common console technology to create a consistent human interface for the autonomic managers of the IT infrastructure. The common console capability provides a framework for reuse and consistent presentation for other autonomic core technologies. The primary goal of a common console is to provide a single platform that can host all of the administrative console functions in server, software, and storage products in a manner that allows users to manage solutions rather than managing individual systems or products.

_ <u>Problem determination :</u> Autonomic managers take actions based on problems or situations they observe in the managed resource. Therefore, one of the most basic capabilities is being able to extract high quality data to determine whether or not a problem exists in managed resources. In this context, a problem is a situation in which an autonomic manager needs to take action. A major cause of poor quality information is the diversity in the format and content of the information provided by the managed resource. To address the diversity of the data collected, a common problem determination architecture normalizes the data collected, in terms of format, content, organization, and sufficiency.

_ <u>Autonomic monitoring :</u> Autonomic monitoring is a capability that provides an extensible run-time environment for an autonomic manager to gather and filter data obtained through sensors. Autonomic managers can utilize this capability as a mechanism for representing, filtering, aggregating, and performing a range of analysis of sensor data.

_ Complex analysis : Autonomic managers need to have the capability to perform complex data analysis and reasoning on the information provided through sensors. The analysis will be influenced by stored knowledge data. An autonomic manager's ability to quickly analyze and make sense of this data is crucial to its successful operation.

_ Policy-based management : Policies are a key part of the knowledge used by autonomic managers to make decisions, essentially controlling the planning portion of the autonomic manager. By defining policies in a standard way, they can be shared across autonomic managers to enable entire systems to be managed by a common set of policies.

_ Heterogeneous workload management : Heterogeneous workload management includes the capability to instrument system components uniformly to manage workflow through the system. Business workload management is a core technology that monitors end-to-end response times for transactions or segments of transactions, rather than at the component level, across the heterogeneous infrastructure.

These capabilities are enabled through a series of tools and facilities that are collected in IBM Autonomic Computing Toolkit. [5]

2.6. Artificial Intelligence and Autonomic Computing

Is autonomic computing basically artificial intelligence? No. Artificial intelligence implies direct conscious thought—a higher level of thinking and self-awareness that rivals human consciousness. Autonomic computing, however, operates below the level of conscious thought—analogous to bodily functions like heart rate, respiratory rate, oxygen levels, and digestion that your body adjusts without your direct control. Autonomic systems are being created in this manner to recognize external threats or internal problems and then take measures to automatically prevent or correct those issues before humans even know there is a problem. These systems are also being designed to manage and proactively improve their own performance, all of which frees IT staff to focus their real—not artificial—intelligence on big-picture projects.

However, Artificial Intelligence and Autonomic Computing have features in common. Mainly both of them target the enablement of machines to take reliable decisions and actions; both result in intelligent systems and thinking machines.

2.7. Examples of Autonomic Systems currently in use

A variety of autonomic computing capabilities are already in use throughout IBM products, and these products are already helping companies succeed. IBM Self-Managing Autonomic Computing capabilities are present in all IBM software product families; Information Management, Lotus, Tivoli, Rational, Websphere, … .

3. Standards overview

3.1. Importance of Standards

A computing environment involves diverse systems working together and connecting to devices and applications across platforms, organizations, and even geographic borders. This environment helps to enable a business to respond quickly to changes in markets, technologies, and the needs of their customers. Businesses must be able to rapidly provide new capabilities to their systems without completely discarding and replacing those applications and systems. [1]

The only way all of these components, applications and systems can work together is with open industry standards. With open standards, businesses and providers can ensure that their products and systems will work and communicate with other systems. [1]

The United States government's National Institute of Standards and Technology (NIST) notes: "Standards are essential elements of information technology-hardware, software, and networks. Standard interfaces, for example, permit disparate devices and applications to communicate and work together. Standards also underpin computer security and information privacy, and they are critical to realizing many widespread benefits that advances in electronic and mobile commerce are anticipated to deliver." [3]

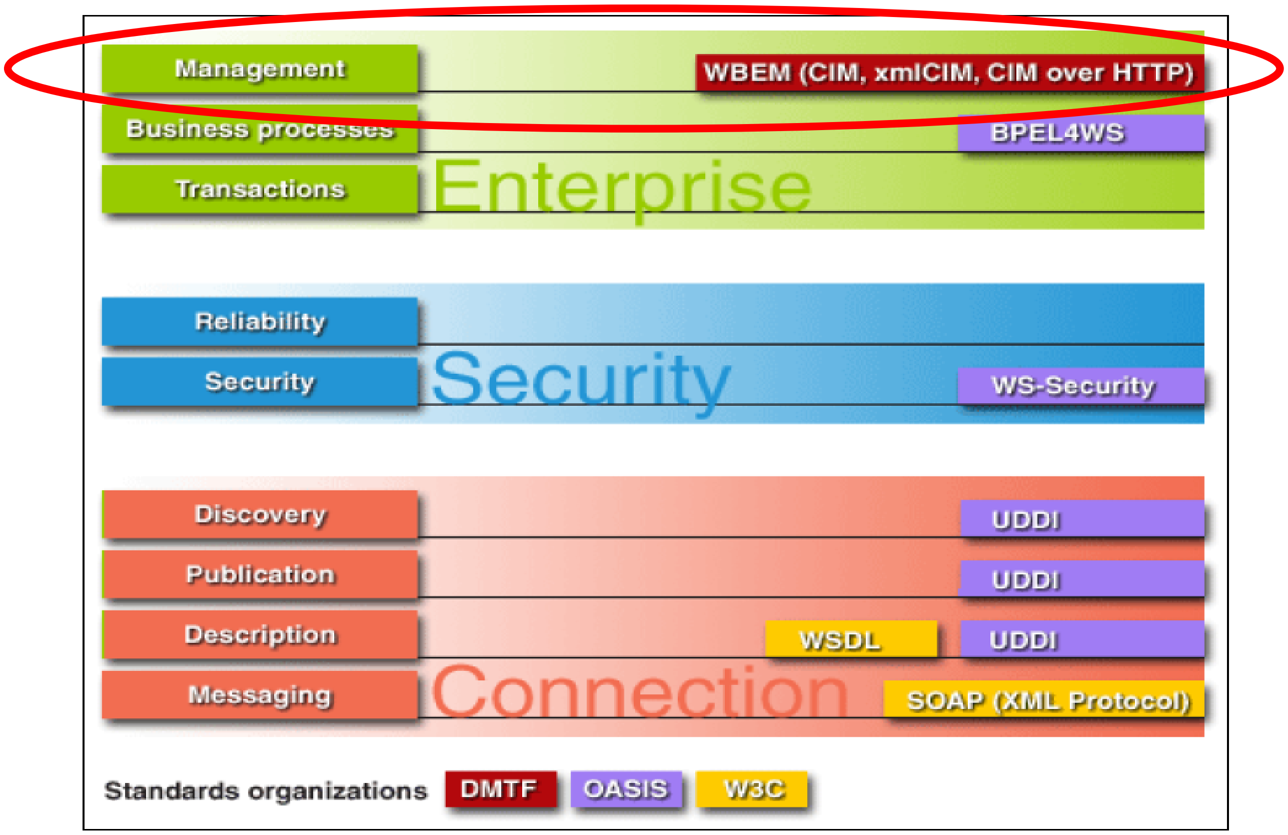


Figure 4. Key Standards

3.2. Standards organizations

DMTF - Distributed Management Task Force

The Distributed Management Task Force, Inc. (DMTF) is a not-for-profit association of industry members dedicated to promoting enterprise and systems management and interoperability. It promotes the development and adoption of interoperable management standards for enterprise and Internet environments. They developed the Common Information Model (CIM) standard, which describes a platform-independent method for exchanging management information. The standard helps to simplify integration and reduce costs of management systems by enabling an end-to-end multi-vendor interoperability. By implementing CIM, vendors and standards groups make possible more integrated and cost-effective management systems. [1]

DMTF standard we are concerned with here is Common Information Model (CIM).

For more information on DMTF, see references[18].

3.3. Key standards

CIM - Common Information Model

CIM is a DMTF standard for expressing data about systems, applications, networks, and devices. It is a conceptual information model for describing computing and business entities in enterprise and Internet environments. It provides a consistent definition and structure of data, using object-oriented techniques. CIM allows various management applications to access the data and control the devices or systems regardless of the platforms involved, making interoperability easier to achieve. [1]

3.4. Perception of Autonomic Computing Standards in Marketplace

Since autonomic IT systems will likely be comprised of multiple components from multiple vendors, it is critical that the various system components be able to communicate with and understand one another. That's why IBM is an active leader in the development of open standards for autonomic computing. This is a significant challenge because of the many different technology areas that autonomic computing spans.

In November 2004, IBM and Fujitsu, a large and influential Japanese IT corporation, announced their agreement to collaborate on autonomic computing standards. This event was part culmination and part catalyst; it was the result of discussions between the two companies about their shared interest in autonomic computing technology and in developing and promoting open standards on which autonomic computing architecture could be based. It also sparked additional interest in autonomic computing technology in Japan, leading to a follow-up announcement about one week later.

According to the press release for the IBM-Fujitsu standards collaboration agreement,

Fujitsu Limited and IBM today announced that they have agreed to collaborate on autonomic computing standards. Initial areas of collaboration are expected to include existing or new standardization efforts related to the Web Services Distributed Management (WSDM) Event Format, standardization of a set of actions to manage IT resources, and standards related to software installation and deployment....

Fujitsu, IBM and other companies have been pursuing autonomic computing technologies on their own, but in order to enjoy the benefits of these technological advances in heterogeneous systems environments, it is critical to promote autonomic computing standards that enable multi-vendor interoperability. Fujitsu's and IBM's new collaboration aims to advance this important effort.

This agreement between two very large IT corporations to collaborate on open standards that are critical to autonomic computing technology was widely publicized, but it was not isolated. A press release from Tokyo on December 6, 2004 noted several additional companies that embrace autonomic computing technologies and announced that they would join IBM's autonomic computing initiative.

These companies joined four others, already announced as IBM autonomic computing business partners, who have already deployed autonomic computing capabilities or standards from the Autonomic Computing Toolkit: Hitachi Software Engineering Co., Ltd, NIWS Co., Ltd., NS Solutions Corporation, and Toshiba Solutions Corporation. The list of Japanese companies who have embraced autonomic computing technology reads like a significant portion of a virtual "Who's Who in Computing in Japan" (including, of course, IBM Japan).

Autonomic computing continues to become more widely known, with increasing adoption in the IT industry. IBM has many autonomic computing business partners from around the globe. Even so, there seems to be an especially sharp interest in autonomic computing in Japan. [6]

4. CIM – Common Information Model

4.1. Overview

The Common Information Model (CIM), currently in version 2.15, provides a data modeling environment in the form of object-like design diagrams and a language-neutral description of the model known as the Managed Object Format (MOF). In this regard, the CIM modeling environment and MOF closely parallel the pairing of the Unified Modeling Language (UML) and Interface Definition Language (IDL).

The CIM provides object classes, properties, methods, and associations common to the use of management applications in the form of management schema. These are organized into three layers:

- Core model : an information model that captures notions that are applicable to all areas of management. While it is possible that additional classes will be added to the Core model over time, major re-interpretations of the Core model classes are not anticipated.
- Common model : an information model that captures notions that are common to particular management areas, but independent of a particular technology or implementation. The common areas are systems, applications, databases, networks and devices. The information model is specific enough to provide a basis for the development of management applications. This model provides a set of base classes for extension into the area of technology-specific schemas. The Core and Common models together are expressed as the CIM schema.
- Extension schemas : represent technology-specific extensions of the Common model. These schemas are specific to environments, such as operating systems (for example, UNIX or Microsoft Windows).

CIM is independent of the method used for implementation, whether it is used in database modeling, in objects design in OOP environment, or in information exchange/parameter passing using a neutral data exchange method like XML representation.

The ability to exchange information between management applications is fundamental to CIM. The current mechanism for exchanging management information is the Managed Object Format (MOF). At the present time, no programming interfaces or protocols are defined by (and hence cannot be considered as) an exchange mechanism. Therefore, a CIM-capable system must be able to import and export properly formed MOF constructs. How the import and export operations are performed is an implementation detail for the CIM-capable system.

The CIM development cycle is very similar to the object-oriented development cycle, in that we'll develop class diagrams, create language-neutral descriptions of the classes and their associations, implement the classes, then deploy and use the resulting data and operations. CIM development mirrors object-oriented development in two important ways: it uses inheritance to extend existing classes, and it uses class diagrams as its primary mechanism for conveying data structure.

Unlike object-oriented classes, however, CIM classes contain keys that uniquely identify object instances. Pure object-oriented design does not use keys for instance identification (though some object technologies -- such as Enterprise JavaBeans -- do have key definitions). In addition, CIM contains a special *association* class that behaves much like a database join. The association relates instances, such as a disk enclosure and the physical disks within that enclosure. Rather than creating relationships within the enclosure and disk classes, the association defines the relationship between the two classes. The classes themselves have no knowledge of their relationship. Instances are navigated through associations rather than by explicit queries to the managed resource class instances. Association navigation is made possible through the use of keys in the CIM object instances.

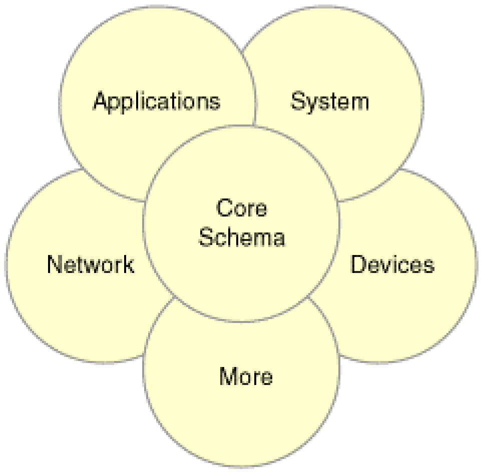
The difference between CIM and pure object-oriented design can be explained by the difference in each environment's intended use. Whereas object-oriented technology has evolved to serve the requirements of creating application programming environments, CIM is specifically suited to describing, cataloging, and interacting with managed resources.

To further understand how the CIM works, we should look at the ways its class hierarchy and various schemas are created and structured.

4.2. How CIM schemas are created

The DMTF goes through iterations in the standards body to create a layered class hierarchy that correctly represents manageable elements from a variety of areas that require management. Each management area, such as storage or applications, is represented in a CIM schema. Different management areas are worked on by different DMTF standards bodies that specialize in those areas. Figure 5 shows how existing CIM schemas are conceptually layered. A core schema is at the center, and schemas then build on each other to represent more specific management areas.

Figure 5. CIM schemas



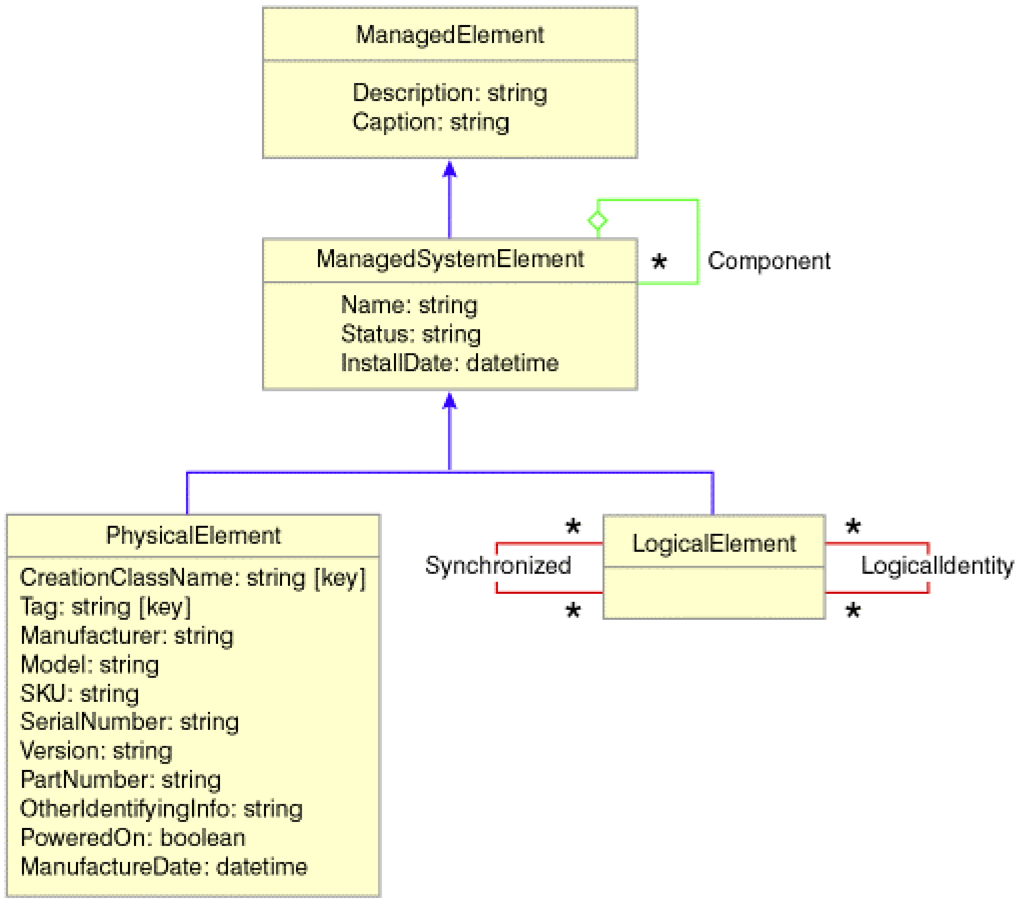
The core schema is structured along the same lines as the Java platform class hierarchy. The core schema contains classes and associations that are common to all of the management areas. For example, all elements that can be managed inherit from a class known as a `CIM_ManagedElement`. In Listing 1, below, we can see that a managed element contains a base set of attributes that are common to all manageable entities.

Listing 1. Class CIM_ManagedElement

```
// ==================================================================
// ManagedElement
// ==================================================================
     [Abstract, Description (
     "ManagedElement is an abstract class that provides a common "
         "superclass (or top of the inheritance tree) for the "
         "non-association classes in the CIM Schema.")]
     class CIM_ManagedElement
     {
       [MaxLen (64), Description (
        "The Caption property is a short textual description (one-"
        "line string) of the object.") ]
       string Caption;
       [Description (
        "The Description property provides a textual description of "
        "the object.") ]
       string Description;
     };
```

Subclasses of the managed element include logical elements and physical elements, as illustrated by Figure 6, a CIM model. A physical element is something that can be touched and/or barcoded. A logical element often represents a software layer.

Figure 6. Base CIM hierarchy

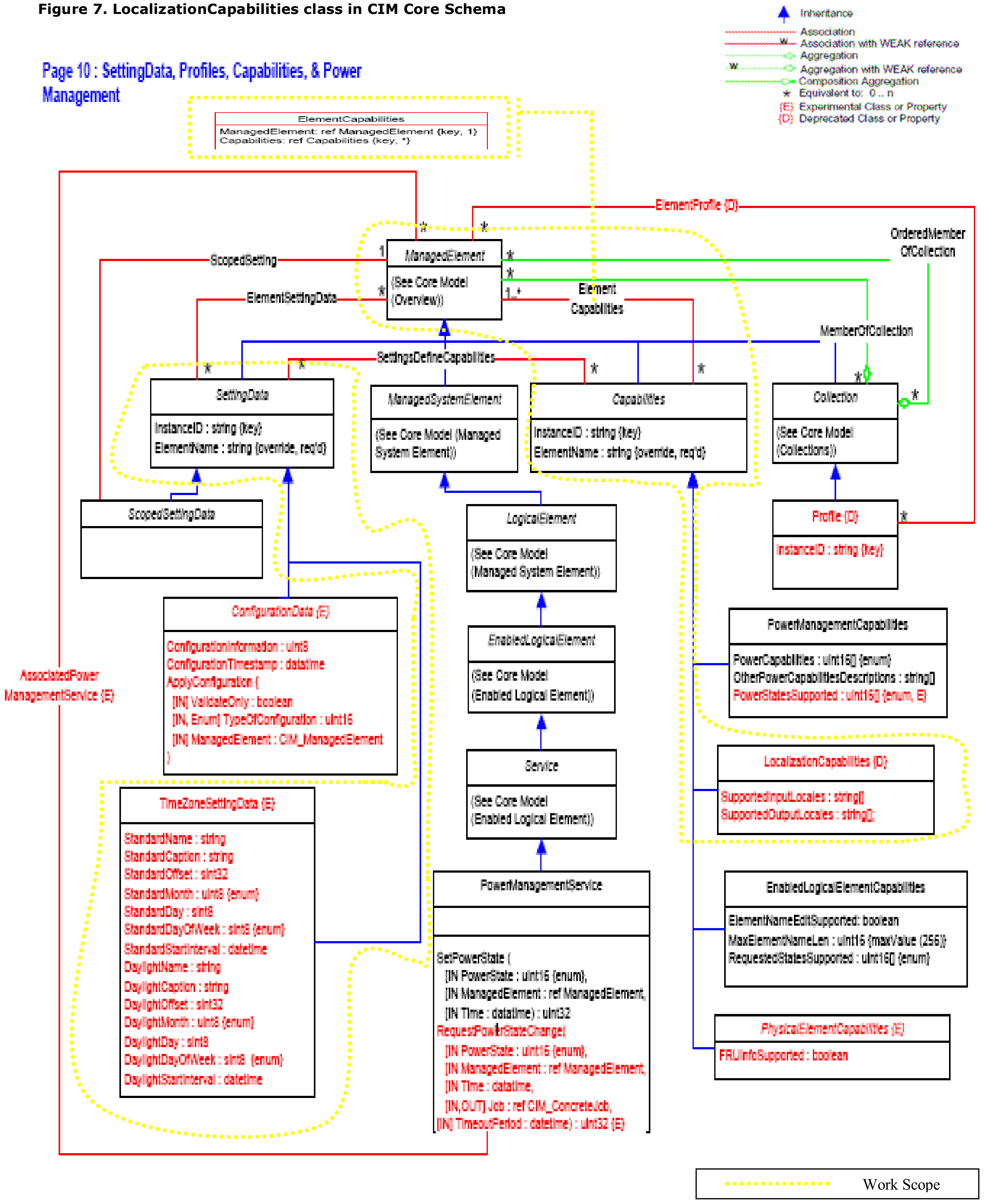


Beyond the core schema, individual working groups within DMTF have created standards for particular fields of management, such as networking, devices, users, applications, and more. These standards extend the core schema and evolve separately from the core schema, though they tend to be consolidated for new versions of completed schemas. Association classes are also a large part of schema definitions. Base associations exist for common relationships between managed elements.

4.3. Current Incorporation of Localization in CIM models (CIM 2.15)

A LocalizationCapabilities class is defined in the core schema, as a subclass of the Capabilities class, which is a type of ManagedElement class, and represents capabilities defined for (associated with) a certain ManagedElement.

Figure 7. LocalizationCapabilities class in CIM Core Schema



Inheritance
Association
Association with WEAK reference
Aggregation
Aggregation with WEAK reference
Composition Aggregation
Equivalent to: 0 .. n
{E} Experimental Class or Property
{D} Deprecated Class or Property
Page 10 : SettingData, Profiles, Capabilities, & Power Management
ElementCapabilities
ManagedElement: ref ManagedElement {key, 1}
Capabilities: ref Capabilities {key, *}
ElementProfile {D}
OrderedMember OfCollection
ScopedSetting
ElementSettingData
ManagedElement
(See Core Model (Overview))
Element Capabilities
MemberOfCollection
SettingsDefineCapabilities
SettingData
InstanceID : string {key}
ElementName : string {override, req'd}
ManagedSystemElement
(See Core Model (Managed System Element))
Capabilities
InstanceID : string {key}
ElementName : string {override, req'd}
Collection
(See Core Model (Collections))
ScopedSettingData
LogicalElement
(See Core Model (Managed System Element))
Profile {D}
InstanceID : string {key}
ConfigurationData {E}
ConfigurationInformation : uint8
ConfigurationTimestamp : datetime
ApplyConfiguration (
[IN] ValidateOnly : boolean
[IN, Enum] TypeOfConfiguration : uint16
[IN] ManagedElement : CIM_ManagedElement
)
EnabledLogicalElement
(See Core Model (Enabled Logical Element))
PowerManagementCapabilities
PowerCapabilities : uint16[] {enum}
OtherPowerCapabilitiesDescriptions : string[]
PowerStatesSupported : uint16[] {enum, E}
AssociatedPower ManagementService {E}
Service
(See Core Model (Enabled Logical Element))
LocalizationCapabilities {D}
SupportedInputLocales : string[]
SupportedOutputLocales : string[]
TimeZoneSettingData {E}
StandardName : string
StandardCaption : string
StandardOffset : sint32
StandardMonth : uint8 {enum}
StandardDay : sint8
StandardDayOfWeek : sint8 {enum}
StandardStartInterval : datetime
DaylightName : string
DaylightCaption : string
DaylightOffset : sint32
DaylightMonth : uint8 {enum}
DaylightDay : sint8
DaylightDayOfWeek : sint8 {enum}
DaylightStartInterval : datetime
PowerManagementService
SetPowerState (
[IN PowerState : uint16 {enum},
[IN ManagedElement : ref ManagedElement,
[IN Time : datatime) : uint32
RequestPowerStateChange(
[IN PowerState : uint16 {enum},
[IN ManagedElement : ref ManagedElement,
[IN Time : datatime,
[IN,OUT] Job : ref CIM_ConcreteJob,
[IN] TimeoutPeriod : datetime) : uint32 {E}
EnabledLogicalElementCapabilities
ElementNameEditSupported: boolean
MaxElementNameLen : uint16 {maxValue (256)}
RequestedStatesSupported : uint16[] {enum}
PhysicalElementCapabilities {E}
FRUInfoSupported : boolean
Work Scope

Listing 2. MOF description of LocalizationCapabilities class

Core\CIM_LocalizationCapabilities.mof

CIM_LocalizationCapabilities	**Superclass: CIM_Capabilities**

Describes the input and output localization capabilities of the entity associated via ElementCapabilities.

Qualifiers: Version ("2.9.0") UMLPackagePath ("CIM::Core::Capabilities")

Parameters (local in grey)

string **SupportedOutputLocales** [ ] ;
This property, along with locale properties in (for example) a *SettingData* class, specifies the locale for data produced by a *ManagedElement.*
A locale indicates a particular geographical, political, or cultural region. The *SupportedOutputLocales* property is an array of strings whose entries specify one or more languages that can be used in the formulation of information requested by and delivered to a client. It may be defined by the *ManagedElement* or client as an input parameter to a method, as a Setting or *SettingData* property, or via other mechanisms. Note that more than one locale may be specified in this array and later selected for output. If multiple locales are selected, information MUST be returned in each language specified, and indicated as supported by this Capabilities instance.

The *SupportedOutputLocales* property publishes an element's support of various locales for output data. The first array entry MUST define the default locale (the natural language associated with the *ManagedElement).* If the *SupportedOutputLocales* property is empty, it is assumed that the default locale is *en_US (English).*
Each array entry consists of three sub-strings, separated by underscores:
- The first sub-string is the language code, as specified in *ISO639.*
- The second sub-string is the country code, as specified in *ISO3166.*
- The third sub-string is a variant, which is vendor specific.
For example, US English appears as: *'en_US_WIN',* where the *'WIN'* variant would specify a Windows browser-specific collation (if one exists). Since the variant is not standardized, it is not commonly used and is generally limited to easily recognizable values *('WIN', 'UNIX', 'EURO', etc.)* used in standard environments. The language and country codes are required; the variant may be empty.

string **SupportedInputLocales** [ ] ;
This property, along with locale properties in (for example) a *SettingData* class, specifies the locale for data consumed by a *ManagedElement.*
A locale indicates a particular geographical, political, or cultural region. The *SupportedInputLocales* property is an array of strings whose entries specify one or more languages that can be used in the formulation of information input by a client. It may be defined by the *ManagedElement* or client as an input parameter to a method, as a Setting or *SettingData* property, or via other mechanisms.

The *SupportedInputLocales* property publishes an element's support of various locales for input data. The first array entry MUST define the default locale (the

natural language associated with the *ManagedElement).* If the *SupportedInputLocales* property is empty, it is assumed that the default locale is *en_US (English).*
Each array entry consists of three sub-strings, separated by underscores:
- The first sub-string is the language code, as specified in *ISO639.*
- The second sub-string is the country code, as specified in *ISO3166.*
- The third sub-string is a variant, which is vendor specific.

For example, US English appears as: *'en_US_WIN',* where the *'WIN'* variant would specify a Windows browser-specific collation (if one exists). Since the variant is not standardized, it is not commonly used and is generally limited to easily recognizable values *('WIN', 'UNIX', 'EURO', etc.)* used in standard environments.
The language and country codes are required; the variant may be empty.

Required
Override ("ElementName")
string **ElementName** ;
The user friendly name for this instance of Capabilities. In addition, the user friendly name can be used as an index property for a search of query. (Note: Name does not have to be unique within a namespace.)

Key
string **InstanceID** ;
Within the scope of the instantiating Namespace, InstanceID opaquely and uniquely identifies an instance of this class. In order to ensure uniqueness within the NameSpace, the value of InstanceID SHOULD be constructed using the following 'preferred' algorithm:
<OrgID>:<LocalID>
Where *<OrgID>* and *<LocalID>* are separated by a colon ':', and where *<OrgID>* MUST include a copyrighted, trademarked or otherwise unique name that is owned by the business entity creating/defining the *InstanceID,* or is a registered ID that is assigned to the business entity by a recognized global authority (This is similar to the *<Schema Name>_<Class Name>* structure of Schema class names.) In addition, to ensure uniqueness *<OrgID>* MUST NOT contain a colon (':'). When using this algorithm, the first colon to appear in *InstanceID* MUST appear between *<OrgID>* and *<LocalID>.*
<LocalID> is chosen by the business entity and SHOULD not be re-used to identify different underlying (real-world) elements. If the above 'preferred' algorithm is not used, the defining entity MUST assure that the resultant *InstanceID* is not re-used across any *InstanceIDs* produced by this or other providers for this instance's NameSpace.
For DMTF defined instances, the 'preferred' algorithm MUST be used with the *<OrgID>* set to *'CIM'.*

MaxLen (64)
string **Caption** ;
The *Caption* property is a short textual description (one- line string) of the object.

string **Description** ;
The *Description* property provides a textual description of the object.

4.4. Work Scope

The CIM Schema establishes a common conceptual framework that describes the managed environment, and attempts to unify and extend the existing instrumentation and management standards. In addition, CIM's goal is to model all the various aspects of the managed environment, not just a single problem space.

Coping with that goal, we propose enhancements and additions to the LocalizationCapabilities class in the Core model, in order to allow it to support the guidelines and requirements of software localization design and development techniques in a practical way. The objective is to allow the standard to provide a model for multilingual localizable managed resources.

5. Software Globalization/Localization

5.1. Definition

One of the key aspects of globalization is the ability to handle multiple languages. As we all know, much of the original development in the field of computers was done in English. But compared to English, most other languages involve a greater degree of computer processing. Many of these languages, such as Chinese, have a very large character set, while others, such as French, have accent marks, and still others, such as Arabic, have bi-directional (left-to-right and right-to-left) input and output.

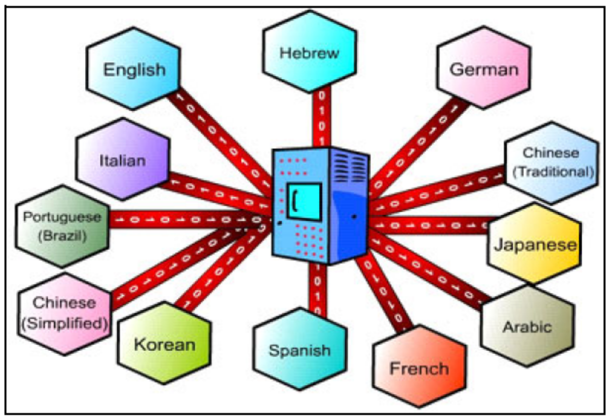


Figure 8. Software Globalization

In information technology, the user interface (UI) is everything designed into an information device with which a human being may interact, including display screen, keyboard, mouse, light pen, the appearance of a desktop, illuminated characters, help messages, and how an application program or a web site invites interaction and responds to it. The user interface can arguably include the total user experience, which may include the aesthetic appearance of the device, response time, and the content that is presented to the user within the context of the user interface.

Localizing the UI of a particular application involves translating the text in localization packs, modifying images and icons as necessary for regional considerations, and modifying layout to accommodate text or image-size changes. [12]

5.2. Locale

In globalization, the word "locale" was borrowed by software engineering from geography to indicate that the distribution of human cultural expectations of computer behavior fall into clumps that can be grouped together, most commonly by language and country or region.

For the purposes of this architecture, a locale means a specification of a language and country/region, or a specification of a language, country/region, and variant. Thus a locale can be specified by a string such as "French-Belgium". It does not mean a data structure that contains information for a language and country/region.

Locale is used by the software industry in general to mean any of the following related concepts:
- The set of people who share a set of common expectations about their computer interactions
- The common expectations of computer behaviors that those people share
- The name given to one of those particular sets of expectations or people
- The computer-readable data (and sometimes code) that encapsulates those behaviors

A locale model contains assumptions about all of these cultural features. In particular, any adequate locale model used in a global e-business system must meet these requirements:
- The locale model accounts for at least language and country/region and has some additional way of specifying variants.
- It includes support for the major categories of locale-dependent computing.
- It provides for hierarchical fall-back behavior at either the source or runtime levels.
- It allows different locales to be set per client. For multi-client server software, this means that there must be a way to have different locale processing for each client context (which may be per thread, depending on the client interaction model).
- The locale model supports conventions that allow all locale-sensitive components of an e-business system to communicate appropriately about locale settings.

Frequently used locale models include : [12]

- Numbers and mathematics

Table 1. Number format

Locale	Positive Format	Negative Format
ar_EG	١٢٬٣٤٥٫٦٧	-١٢٬٣٤٥٫٦٧
de_DE	12.345,67	-12.345,67
en_US	12,345.67	-12,345.67
zh_CN	12,345.67	-12,345.67
fr_CH	12 345,67	-12 345,67
de_CH	12'345,67	-12'345,67
it_CH	12 345,67	-12 345,67

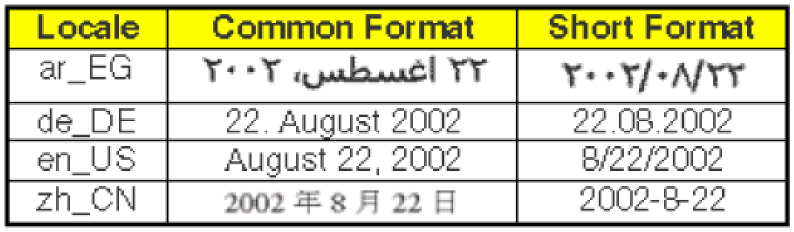
- Currency format

Table 2. Currency format

Locale	Currency Name	Subunits	Positive Format	Negative Format	ISO4217 Code
ar_EG	Egyptian Pound	Piaster, 100 Piasters	١٢٬٣٤٥٫٦٧ ج.م.	-١٢٬٣٤٥٫٦٧ج.م.	EGP
en_US	US Dollar	Cent, 100 Cents	$1,234,567.89	-$1,234,567.89	USD
zh_CN	Yuan Renminbi	1Jiao/10 Fen, 10 Jiao/100 Fen	¥1,234,567	-¥1,234,567	CNY

- Date

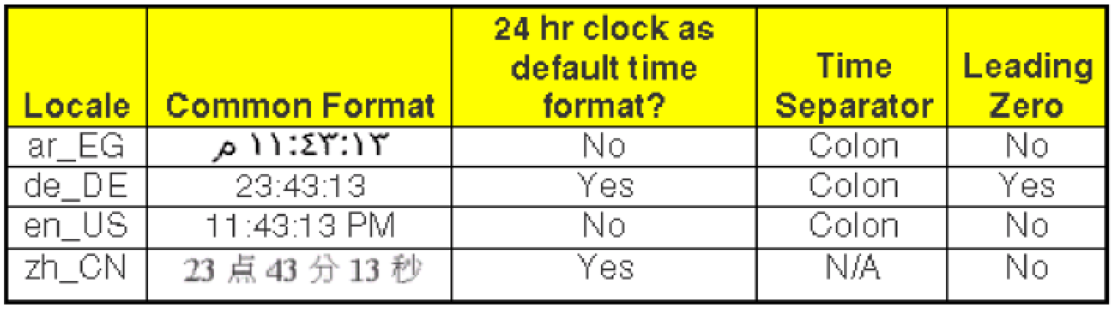
Table 3. Date format



Locale	Common Format	Short Format
ar_EG	٢٢ اغسطس، ٢٠٠٢	٢٠٠٢/٠٨/٢٢
de_DE	22. August 2002	22.08.2002
en_US	August 22, 2002	8/22/2002
zh_CN	2002 年 8 月 22 日	2002-8-22

- Time

Table 4. Time format



Locale	Common Format	24 hr clock as default time format?	Time Separator	Leading Zero
ar_EG	١١:٤٣:١٣ م	No	Colon	No
de_DE	23:43:13	Yes	Colon	Yes
en_US	11:43:13 PM	No	Colon	No
zh_CN	23 点 43 分 13 秒	Yes	N/A	No

- Calendar

- Telephone

Table 5. International Telephone Codes

Country/Region	International Access Code	CCITT/ITU Code	Internal Phone Format
Egypt	00	20	(12) 3456789
Germany	00	49	12345-6789012345678
United States	011	1	(123) 456-7890
China	00	86	(10)65391188

- Measures

- Icons

- conventions (abbreviations, question marks, percent symbol, …)

- Sorting Order

Advanced cultural support might involve business logic. For example, income tax calculators must reflect tax amounts based on nation-specific income policies and the individual's reported income.

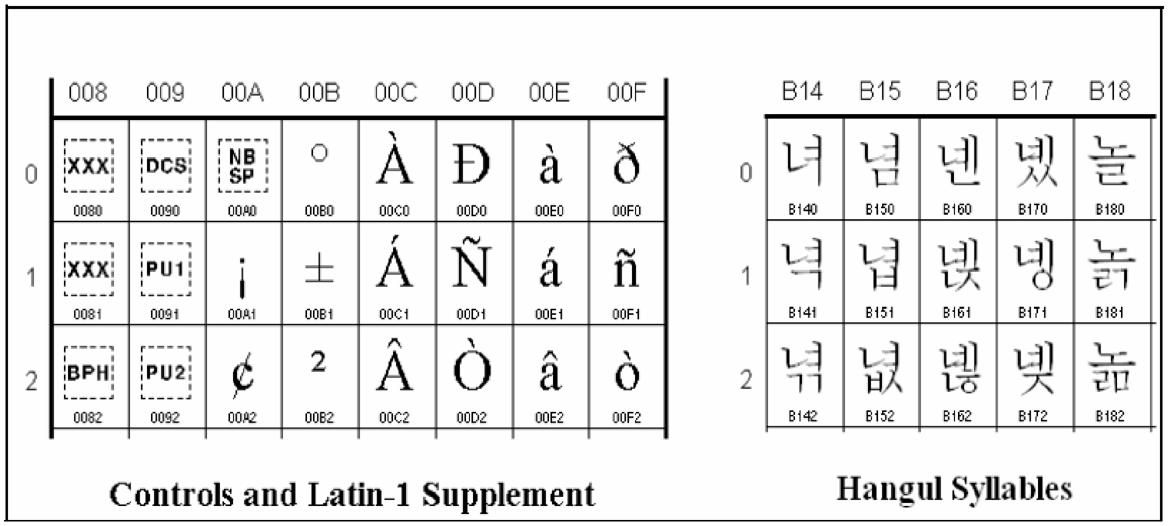
Typically, programs call international services such as those found in Java or ICU to handle the complications found in all of these.

5.3. Unicode support

An encoding system is a method of assigning numbers to individual characters so that a computer can process those characters. In computing's early days, there were hundreds of different encoding systems spanning many different language sets. No single system existed that could process every single character from all languages throughout the world. Another drawback was that those encoding systems might conflict with one another in that the same character could have different "number" representations in different systems. Then Unicode evolved.

Unicode is the universal character-encoding scheme for written characters and text, including character sets used by many of the world's written scripts. By providing a consistent way for handling multilingual text interchange internationally, Unicode is in widespread use today. It has been widely accepted as the default encoding for many industry standards such as HTML and XML that enable Java's capabilities for multilingual support, thus providing one of the principal foundations of e-business. Now when the world wants to talk, it speaks in Unicode.

Unicode provides a unique number for every character, no matter what the platform, no matter what the program, no matter what the language. Unicode can represent every character in the world by providing a single consistent "character to number" mapping schema. [12]



	008	009	00A	00B	00C	00D	00E	00F
0	XXX 0080	DCS 0090	NB SP 00A0	° 00B0	À 00C0	Ð 00D0	à 00E0	ð 00F0
1	XXX 0081	PU1 0091	¡ 00A1	± 00B1	Á 00C1	Ñ 00D1	á 00E1	ñ 00F1
2	BPH 0082	PU2 0092	¢ 00A2	² 00B2	Â 00C2	Ò 00D2	â 00E2	ò 00F2

Controls and Latin-1 Supplement

	B14	B15	B16	B17	B18
0	녀 B140	념 B150	녠 B160	녰 B170	놀 B180
1	녁 B141	녑 B151	녡 B161	녱 B171	놁 B181
2	녂 B142	녒 B152	녢 B162	녲 B172	놂 B182

Hangul Syllables

Figure 9. A single consistent "character-to-number" mapping schema

The International Components for Unicode (ICU) and Java are the recommended ways to handle Unicode text.

5.4. CLDR (Common Locale Data Repository)

A Relatively new project started in 2004, hosted by Unicode Consortium. It aims at achieving the goals of being a repository of common, necessary software locale data for all world languages, and to collect and maintain locale data. It uses XML format for effective interchange. It's Freely available. CLDR is by far the largest and most extensive standard repository of locale data. This data is used by a wide spectrum of companies for their software internationalization and localization. [24]

5.5. ICU (International Components for Unicode)

A cornerstone of the globalization technologies is ICU (International Components for Unicode), an open-source project that serves up cultural data. ICU is a pair of libraries (one in C/C++ and the other in Java – ICU4J) that make it easy to add robust Unicode and internationalization support to various applications. This library provides: [12]

- Calendar support
- Message formatting
- Character set conversions
- Normalization
- Collation (language-sensitive)
- Number and currency formatting
- Date and time formatting
- Time zones
- Locales (200+ supported)
- Transliteration
- Resource bundles
- Word, line, and sentence breaks

5.6. Examples of Localization Impact in Practice

Ex. 1 : Intel Website : IT Flex Services Makes an Impact Worldwide

When the Intel Customer Support (ICS) Web site received an award for being one of the ten best international Web support sites by the Localization Industry Standards Association and the Association of Support Professionals, it was due in no small part to the work of IT Flex Services. As part of their language pilot project, ICS contacted IT Flex Services to localize key documents into 10 different languages. About 750 of the 24,000 documents on the ICS site need updating each week, and the site grows by about 500 documents a month. "It was a unique, synergistic partnership between ICS, as the customer, and the IT Flex Services team," said Lew Tarnopol, ICS program manager, praising the IT Flex Services localization team.

Localizing the Web site reduced traffic to Intel's Customer Support Service Center. It also directly impacts sales, as customers can get the information they need in their own languages. It's part of being a truly global company. [http://www.intel.com/it/performance-report/operational-excellence/it-flex-services.htm]

Ex. 2 : Intel Video & Internet Technology: Smart Media Project

Description:
Develop Smart Media algorithms and demo applications for media management, browsing, editing, smart summarization, and an Internet media publishing system. So far, we have concentrated our efforts on SceneGrid, a video management, browsing, editing, smart summarization, and Internet media publishing system.

- Media publishing system
- Automatic abstraction of raw video footage such home video into shorter videos
- Text localization in images and videos

[http://www.intel.com/technology/computing/applications/smart_media.htm]

Ex. 3 : Windows 2000 : Enhanced Globalization Features

Globalization - or the modification of software for localization in specific geographical areas or dialects - is very important if you intend to increase your global account opportunities. Although not all user interfaces warrant globalization, it is likely that some of your user interfaces (UI) should be localized. The pressure to do this will grow as more software developers take advantage of the tools in products like Windows 2000 and as the customer base grows used to this luxury. For example, most educated technologists can read and write English adequately enough to use an English-based UI for system administration. However, less skilled workers who usually make up call center agent pools, and even your customer's customers, may require some degree of localization for the user interfaces that they come in contact with.

Microsoft has greatly enhanced the localization features and tools that come with Windows 2000. You can start right away to take advantage of these facilities.

[http://www.intel.com/network/csp/resources/white_papers/5871web.htm]

Ex. 4 : Linux : What Does Linux Have to Offer Technically? What's in It for Me? Enhanced Globalization Features

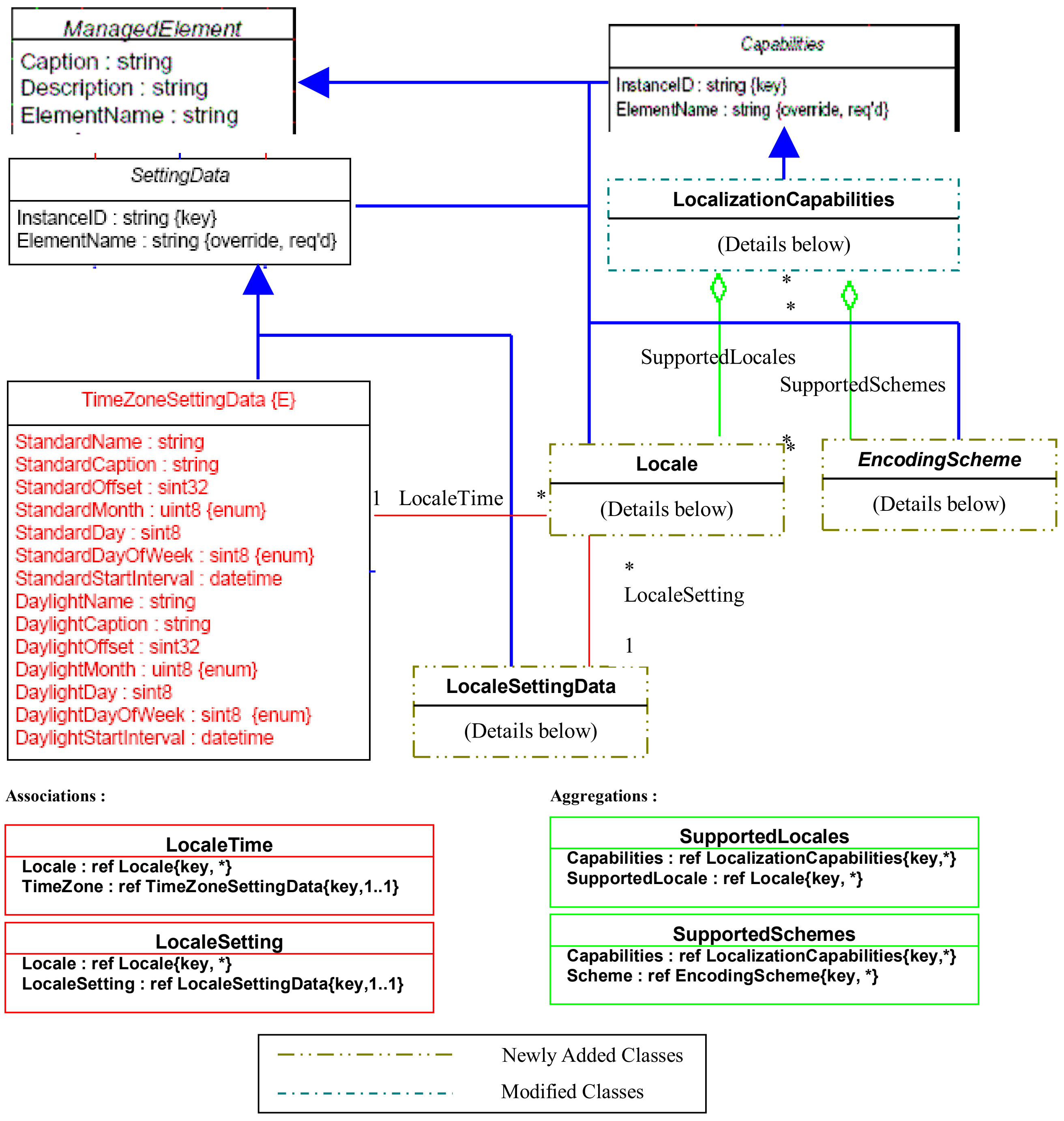
If you're looking to increase your global account opportunities, then globalization, or the modification of software for localization in specific geographical areas or dialects, becomes important. Although not all user interfaces warrant globalization, it is likely that some need to be localized. The pressure to modify software for local use will grow more as software developers make the most of the tools in products like Linux, and as customers get used to this capability. For example, although English is the most commonly used user interface for system administration, solutions often benefit when end-user interfaces are localized. With Linux, you can build on such features right away.

["Assessing Migrating to the Linux Operating System for Converged Communications Solutions" – White Paper]

6. Proposed Model for "LocalizationCapability" Class

6.1. Model Modifications

Models are abstractions of "real world" objects and events. In this notion, the proposed model modifications abstract the localization objects and events in an operating environment. Data flow and configuration perspectives are taken in consideration. Changes made preserve the model backwards compatibility.



ManagedElement
Caption : string
Description : string
ElementName : string
Capabilities
InstanceID : string {key}
ElementName : string {override, req'd}
SettingData
InstanceID : string {key}
ElementName : string {override, req'd}
LocalizationCapabilities
(Details below)
*
*
SupportedLocales
SupportedSchemes
TimeZoneSettingData {E}
StandardName : string
StandardCaption : string
StandardOffset : sint32
StandardMonth : uint8 {enum}
StandardDay : sint8
StandardDayOfWeek : sint8 {enum}
StandardStartInterval : datetime
DaylightName : string
DaylightCaption : string
DaylightOffset : sint32
DaylightMonth : uint8 {enum}
DaylightDay : sint8
DaylightDayOfWeek : sint8 {enum}
DaylightStartInterval : datetime
1
LocaleTime
*
*
*
Locale
(Details below)
EncodingScheme
(Details below)
*
LocaleSetting
1
LocaleSettingData
(Details below)
Associations :
LocaleTime
Locale : ref Locale{key, *}
TimeZone : ref TimeZoneSettingData{key,1..1}
LocaleSetting
Locale : ref Locale{key, *}
LocaleSetting : ref LocaleSettingData{key,1..1}
Aggregations :
SupportedLocales
Capabilities : ref LocalizationCapabilities{key,*}
SupportedLocale : ref Locale{key, *}
SupportedSchemes
Capabilities : ref LocalizationCapabilities{key,*}
Scheme : ref EncodingScheme{key, *}
Newly Added Classes
Modified Classes

Figure 10. Proposed Model for "LocalizationCapabilities" Class

6.2. Proposed Changes

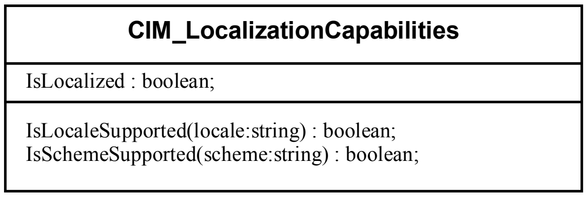
1. Added a new class “Locale” to hold data and functionality required to represent a certain locale. Specific locales supported by the localization capabilities of a certain managed element are instances of this class.
2. Added a new class “LocaleSettingData” as a child of the “SettingData” abstract class, to hold data required to represent the settings of a certain locale.
3. Added a new abstract class “EncodingScheme”, subclasses of which will represent a specific Character Encoding Scheme (Character Set) supported by the localization capabilities of a managed element.
4. Added a new association between the “Locale” and the “TimeZoneSettingData” class, called “LocaleTime”, to represent the relation between a certain locale and its time settings. It is a one-to-many relation since every locale should have one time setting, while a specific time setting may be common between zero or more supported locales.
5. Added a new association between the “Locale” and the “LocaleSettingData” class, called “LocaleSetting”, to represent the relation between a certain locale and its specific settings. It is a one-to-many relation since every locale should have one settings set, while a specific locale setting may be common between zero or more supported locales.
6. Added a new aggregation between the “Locale” class and the “LocalizationCapabilities” class, called “SupportedLocales”, to represent the list of locales supported by the Localization Capabilities of a certain managed element. It should be many-to-many relation, since the localization capabilities of a certain managed element may support zero or more locales, and a specific locale may be supported by zero or more localization capabilities objects.
7. Added a new aggregation between the “EncodingScheme” class and “LocalizationCapabilities” class, called “SupportedSchemes”, to represent the list of Character Encoding Schemes supported by the Localization Capabilities of a certain managed element. It should be many-to-many relation, since the localization capabilities of a certain managed element may support zero or more encoding schemes, and a specific encoding scheme may be supported by zero or more localization capabilities objects.

The CIM Schema is a "keyed" object model. All class instances are uniquely named, and referenced by the class' keys. Associations are uniquely identified by their keys, which have always included their reference properties. All concrete classes must define or inherit a key structure. If inherited, the key structure cannot be changed.

6.3. Classes Details and Description

CIM_LocalizationCapabilities

This class determines the possibilities for a certain managed element to be localized. It holds the required data fields to show its capabilities. It provides as its interface the required methods to communicate with other classes that need to work with the localization of this managed element. The original class elements before the proposed changes contained two string arrays to represent the supported input locales and supported output locales as string literals. These two lists were merged in one array called SupprtedLocales, which can be navigated through the SupportedLocales association object. The rationale behind the merge into one list of all supported locales is that it makes more sense for a certain managed element to be able to communicate with an external element using a certain locale in the two directions, rather than being able to receive input in one locale but incapable of responding with output in this locale. Logically, the two lists of input and output locales should be identical, leading to substitution with one list. In addition, the class is associated with a list of SupportedSchemes, which can be navigated through the SupportedSchemes association object.


CIM_LocalizationCapabilities
IsLocalized : boolean;
IsLocaleSupported(locale:string) : boolean;
IsSchemeSupported(scheme:string) : boolean;

MOF representation

```
[Description("Class representing the localization capabilities of a certain
managed element")]
      class CIM_LocalizationCapabilities : CIM_Capabilities
      {
      [Description(
            "Boolean to indicate if the referred to managed element"
            "supports localization or not. If this flag is false, then all"
            "other features in this class and related classes are set off")]
      boolean IsLocalized;

      [Description(
            "This method indicates whether a certain locale name is"
            "supported by the localization capabilities of a specific"
            "managed element")]
      uint32 IsLocaleSupported([IN] string locale, [OUT] boolean
IsLocalized);

      [Description(
            "This method indicates whether a certain character encoding"
            "scheme is supported by the localization capabilities of a"
            "specific managed element")]
      uint32 IsSchemeSupported([IN] string scheme, [OUT] boolean
SchemeSupport);
      };
```

CIM_Locale

Class to represent every specific supported locale. It contains references to the time zone setting object and the locale setting object which define the several settings specific to this locale. References to TimeZoneSettingData object and LocaleSettingData object can be accessed through the corresponding association objects. Every instance of this class should define the referenced objects (TimeZoneSettingData and LocaleSettingData) according to its particular requirements. The class contains an identifier, LocaleIdentifier, to hold the name of the standards body used to load the LocaleSettingData and TimeZoneSettingData fields' values at instantiation time, default value is 'CLDR'. The modeling of the used standard in this manner gives more flexibility for support of new standards that may be emerging in the future.

```
CIM_Locale
--------------------------------------------------------------------------------
FlowDirectionsOrder : string[] {enum, read};  // rtl, ltr, tb-rl, lr-tb, rl-tb
CourtesyTitlesOrder : string[] {enum, read}; // Ms., Mrs., Mr., Sir, Dr., Prof., …
GreetingsOrder : string[] {enum, read}; // G.M., GAN, G.N., Welcome, Bye, …
MeasureUnitsOrder : string[] {enum, read}; // Weight, Length, Freq., Temp., …
PunctuationSymbolOrder : char16[] {enum, read}; // , . ; / \ : " ' …..
VowelsOrder : string[] {enum, read}; // a, i, u, e, o, aa, ii, uu, ee, oo, stress, consonant
LocaleIdentifier : string;
LocaleName : string;
```

MOF representation

```
[Description("Class representing the system locale")]
      class CIM_Locale : CIM_ManagedElement
      {
      [read, Description(
            "Array of directional abbreviations. LocaleSettingData objects"
            "will support text flow directions based on this order"),
            Values{“rtl”, “ltr”, “tb-rl”, “lr-tb”, “rl-tb”}]
      String[] FlowDirectionsOrder;

      [read, Description(
            "Array of courtesy titles. LocaleSettingData objects will"
            "provide locale specific titles based on this order"),
            Values{“Ms.”, “Mrs.”, “Mr.”, “Sir”, “Dr.”, “Prof.”}]
      String[] CourtesyTitlesOrder;

      [read, Description(
            "Array of greetings. LocaleSettingData objects will provide"
            "locale specific greetings based on this order"),
            Values{“G.M.”, “GAN”, “G.N.”, “Welcome”, “Bye”}]
      String[] GreetingsOrder;

      [read, Description(
            "Array of measure units. LocaleSettingData objects will"
            "provide locale specific units based on this order"),
            Values{“Weight”, “Length”, “Freq.”, “Temp.”}]
      String[] MeasureUnitsOrder;

      [read, Description(
            "Array of punctuation symbols. LocaleSettingData objects will"
            "provide locale specific symbols based on this order"),
            Values{“,”, “.”, “;”, “/”, “\”, “:”, “”””, “‘”}]
      char16[] PunctuationSymbolOrder;

      [read, Description(
            "Array of vowels. LocaleSettingData objects will provide"
            "locale specific vowels based on this order"),
            Values{“a”, “i”, “u”, “e”, “o”, “aa”, “ii”, “uu”, “ee”, “oo”,
            “stress”, “consonant”}]
      String[] VowelsOrder;

      [Description(
            "The standard body used to fill the locale setting data at"
            "instantiation time. Default value is ‘CLDR’"]
      string LocaleIdentifier;

      [Description(
            "String to hold the name designating this locale")]
      string LocaleName;
      };
```

CIM_LocaleSettingData

This class is intended to keep the specific settings for a certain locale in its fields. It's a subclass of the abstract class "SettingData" in the core model. Fields of this class are visible to the "Locale" class, and should be set according to the specific locales they represent, either by loading their values from the CLDR locale repository at instantiation time (Or other locale identifier, if any, as specified in LocaleIdentifier attribute in Locale class), or by setting them individually according to specific needs using, for example, resource bundles. The same way of setting values for attributes of this class is used as well for setting values for the attributes of the TimeZoneSettingData object associated with a specific locale object.

CIM_LocaleSettingData

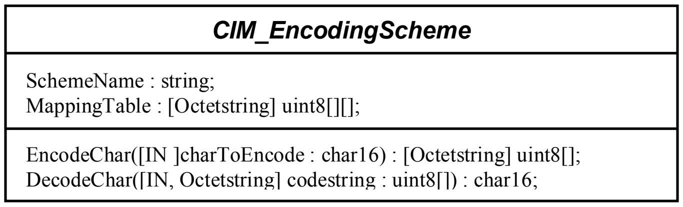
```
Country : string;
Language : string;
Script : string;
bidiSupport : boolean;
AnnotationSupport : boolean;
VowelizationSupport : boolean;
CurrencyName : string;
CurrencySymbol : char16;
DateFormates : string[]; // Holds format strings available in this locale
// Flags corresponding to the entries in the FlowDirectionsOrder enum
FlowDirections : boolean[];
// Entries in this locale corresponding to the entries in enum data in Locale
PunctuationSymbols : char16[];
CourtesyTitles : string[];
Greetings : string[];
MeasureUnits : string[];
Vowels : char16[];
// Map character to its transliteration, array index in the character hash value
CharTransliteration :string[]; // In English script
```

MOF representation

```
[Description("Class representing the settings of a specific system locale")]
      class CIM_LocaleSettingData : CIM_SettingData
      {
      [Description(
            "String to hold the name of the country in which the"
            "locale associated with this LocaleSettingData object is"
            "prevailing")]
      string Country;

      [Description(
            "String to hold the language used in the locale associated"
            "with this LocaleSettingData object")]
      string Language;

      [Description(
            "String to hold the name of the writing script used with the"
            "locale associated with this LocaleSettingData object")]
      string Script;

      [Description(
            "Flag to show if the locale associated with this"
            "LocaleSettingData object supports bidirectional text or not")]
      boolean bidiSupport;

      [Description(
            "Flag to show if the locale associated with this"
            "LocaleSettingData object supports text annotation – like"
            "Japanese - or not")]
      boolean AnnotationSupport;

      [Description(
            "Flag to show if the locale associated with this"
            "LocaleSettingData object supports character vowelization"
            "signs – like Arabic and Hebrew - or not")]
      boolean VowelizationSupport;

      [Description(
            "The name of the currency used in the locale associated with"
            "this LocaleSettingData object")]
      string CurrencyName;

      [Description(
            "The symbol of the currency used in the locale associated with"
            "this LocaleSettingData object")]
      char16 CurrencySymbol;

      [Description(
            "Holds date format strings available in this locale")]
      string[] DateFormates;

      [Description(
            "Flags corresponding to the entries in the FlowDirectionsOrder"
            "enum object. A true value in a certain position means that"
            "this text flow direction is supported.")]
      boolean[] FlowDirections;
      [Description(
            "Entries in this locale corresponding to the entries in the"
            "PunctuationSymbolOrder enum object in Locale. Encoded"
            "in UTF-8 by default")]
      char16[] PunctuationSymbols;

      [Description(
            "Entries in this locale corresponding to the entries in the"
            "CourtesyTitlesOrder enum object in Locale. Encoded in"
            "UTF-8 by default")]
      string[] CourtesyTitles;

      [Description(
            "Entries in this locale corresponding to the entries in the"
            "GreetingsOrder enum object in Locale. Encoded in UTF-8"
            "by default")]
      string[] Greetings;

      [Description(
            "Entries in this locale corresponding to the entries in the"
            "MeasureUnitsOrder enum object in Locale. Encoded in"
            "UTF-8 by default")]
      string[] MeasureUnits;

      [Description(
            "Entries in this locale corresponding to the entries in the"
            "VowelsOrder enum object in Locale. Encoded in UTF-8 by"
            "default")]
      char16[] Vowels;

      [Description(
            "Map every character in this locale to its transliteration in"
            "English script, array index is the character hash value")]
      string[] CharTransliteration;
};
```

CIM_EncodingScheme

Each subclass of this abstract class will represent an encoding scheme used by the localization capabilities of a specific managed element to represent characters of supported locales. This includes encoding schemes, among others, like **UTF-8, UTF-16, UTF-16BE, UTF-16LE, UTF-32, ISO_8859-1:1987, ISO_8859-1, ISO-8859-1, latin1,L1, IBM819, CP819, csISOLatin1, Windows-1252, GB2312, BIG5, BIG5-HKSCS, SHIFT_JIS, HP-Legal, ISO 8859-1, IBM 850, ISO-2022-JP, Shift_JIS, EUC-JP (JIS X-0208-1997). [Details on Character Encoding Schemes can be found at IANA (Internet Assigned Numbers Authority) : www.iana.org/assignments/character-sets]**. Actual implementation of subclasses of this class may use library methods for encoding/decoding using a certain standard encoding scheme.



CIM_EncodingScheme

SchemeName : string;
MappingTable : [Octetstring] uint8[][];

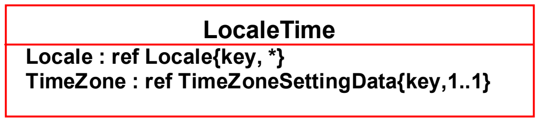
EncodeChar([IN]charToEncode : char16) : [Octetstring] uint8[];
DecodeChar([IN, Octetstring] codestring : uint8[]) : char16;

MOF representation

```
[Abstract, Description("Class representing the encoding schemes")]
class CIM_EncodingScheme : CIM_ManagedElement
{
      [Description(
            "Scheme Name")]
      string SchemeName;

      [Octetstring, Description(
            "Map a character to its code value, array index in the"
            "character hash value. No need for the MappingTable in"
            "case of usage of library methods for"
            "encoding/decoding. It’s more useful for custom"
            "encoding/decoding schemes.")]
      uint8[][] MappingTable;

      [Octetstring, Description (
            "Return the code of the given character in this"
            "encoding scheme. Default UTF-8 for parameter character"
            "passing")]
      uint32 EncodeChar([IN] char16 charToEncode, [OUT] uint8[] code);

      [Description (
            "Return the character given its code in this encoding"
            "scheme. Default UTF-8 for parameter character passing")]
      uint32 DecodeChar([IN, Octetstring] uint8[] codeString, [OUT]
char16 char);
};
```

CIM_LocaleTime

This association represents the time zone settings of a specific locale.

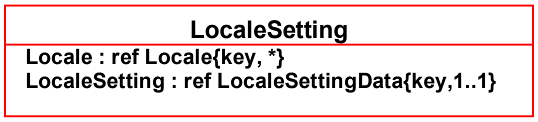

LocaleTime
Locale : ref Locale{key, *}
TimeZone : ref TimeZoneSettingData{key,1..1}

MOF representation

```
[Association, Description("The time zone settings associated with a
specific locale")]
class CIM_LocaleTime : CIM_Dependency
{
      [key, Override ("Locale"),
            Description ("The specific Locale object")]
      Locale ref Locale;

      [key, Override ("TimeZone"),
            Description (
                  "Time zone settings corresponding to the"
                  "specified locale ")]
      TimeZoneSettingData ref TimeZone;
};
```

CIM_LocaleSetting

This association represents the settings data of a specific locale.

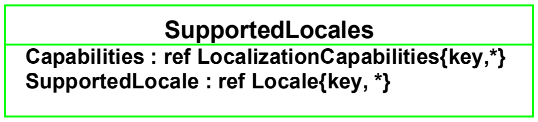

LocaleSetting
Locale : ref Locale{key, *}
LocaleSetting : ref LocaleSettingData{key,1..1}

MOF representation

```
[Association, Description("Locale specific setting data")]
class CIM_LocaleSetting : CIM_Dependency
{
      [key, Override ("Locale"),
            Description ("The specific Locale object")]
      Locale ref Locale;

      [key, Override ("LocaleSetting"),
            Description (
                  "The settings related to the specified locale")]
      LocaleSettingData ref LocaleSetting;
};
```

CIM_SupportedLocales

This association represents the system locales aggregated by the capabilities of a certain managed element.

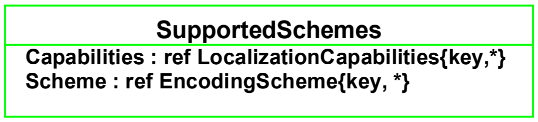

SupportedLocales
Capabilities : ref LocalizationCapabilities{key,*}
SupportedLocale : ref Locale{key, *}

MOF representation

```
[Association, Aggregation, Description("List of system locales of
which a certain managed element is capable")]
class CIM_SupportedLocales : CIM_ConcreteComponent
{
      [key, Override ("Capabilities"), Aggregate,
            Description (
                  "The capabilities object of a certain managed"
                  "element")]
      LocalizationCapabilities ref Capabilities;

      [key, Override ("SupportedLocale"),
            Description (
                  "The locales supported by the specific managed"
                  "element capabilities")]
      Locale ref SupportedLocale;
};
```

CIM_SupportedSchemes

This association represents the list of encoding schemes aggregated by the capabilities of a certain managed element.


SupportedSchemes
Capabilities : ref LocalizationCapabilities{key,*}
Scheme : ref EncodingScheme{key, *}

MOF representation

```
[Association, Aggregation, Description("List of encoding schemes of
which a certain managed element is capable")]
class CIM_SupportedSchemes : CIM_ConcreteComponent
{
      [key, Override ("Capabilities"), Aggregate,
            Description (
                  "The capabilities object of a certain managed"
                  "element ")]
      LocalizationCapabilities ref Capabilities;

      [key, Override ("Scheme"),
            Description (
                  "The schemes supported by the specific managed"
                  "element capabilities ")]
      EncodingScheme ref Scheme;
};
```

6.4. Significance of Modeling Localization

Choice of the LocalizationCapabilities class enhancement as our work scope originated from realization of the importance and vitality of localization of several systems in a global flat world. Localization/Globalization is an inevitable requirement in all kinds of industry. This doesn't refer only to language difference, but also to different geographical and cultural aspects. Market hunting in any area/country depends in a big deal on the ability to understand these differences in people nature and to communicate with them according to this understanding, in a way that fits their correct perception. That's why business globalization is a must in today's world.

The following excerpt from LISA (Localization Industry Standards Association) FAQs emphasizes this idea :

Why does globalization matter to different companies?

If a company does business in more than one country, it is already facing globalization issues, whether they are aware of them or not. If globalization is not thought through and planned for, costs go up while quality and customer satisfaction decline. If a company has an Internet site, it already has international exposure. Globalization represents the opportunity to capitalize on that exposure.

In addition, domestic markets are now rarely free from international competition and are seen as major growth opportunities for companies that have reached a plateau in growth within their home markets. Many companies make in excess of 70% of their revenues outside of their home markets, and these levels are not restricted to just large IT companies, but are found by smaller, "niche" companies as well.

What economic impact does globalization have?

Globalization provides a way for enterprises to sell products and services in other markets, enabling them to grow beyond what would be possible in their home markets. Many of the leading companies in the Global Fortune 500 operate at a profit only because they are globalized, and LISA estimates that every dollar spent on globalization unlocks roughly $25 of additional revenue for companies engaged in globalization.

Globalization is not just a way for European and North American companies to sell products elsewhere; it also provides a way for growing companies in emerging economies to access developed markets. By increasing the places from which products can come, globalization promotes improved quality of service and decreased costs for consumers.

More explanation of importance and significance of localization modeling in CIM model can be deduced from the following excerpt quoted from "CIM Diagnostics Model White Paper" :

"Localization refers to the support of various geographical, political, or cultural region preferences, or locales. A client may be in a different country from the system it is querying and would prefer to be able to communicate with the system using its own locale. There are inherent differences to be reckoned with, such as language, phraseology, currency, and many cultural oddities.

There is no localization support in CIM prior to Version 2.9. Since diagnostics relies on precise reporting of system status and problem data in a user-centric environment, localization is critical. In V2.9, we introduced schema extensions to allow a client to query a diagnostic service for supported locales and to specify the desired locale via a DiagnosticSetting object. The change was written as generically as possible, specifically supporting diagnostics with the intent that it be generalized for broader use in the future.

A new class, CIM_LocalizationCapabilities : CIM_Capabilities was introduced with properties publishing the supported input and output locals. A Locales[] property is added to the DiagnosticSetting class (for passing to the service) and the DiagnosticServiceRecord class (for local identification of the resultant logs)."

<u>Rationale behind class deprecation and applicability of proposed reactivation can be explained by inspection of the stated reasoning in the change request issued in CIM V2.10, as quoted below :</u>

"There was a discussion at the TC regarding the modeling of localization. Preference is to do localization in the protocol (CIM/XML over HTTP) rather than in the schema. This CR deprecates the localization schema to favor usage of the HTTP Accept-Language and Content-Language header fields as documented in DSP0200, Specification for CIM Operations over HTTP v1.2." [Related excerpts from DSP0200 can be found in appendices].

It's clear that this change request put an assumption that the localization capabilities are limited to different languages support, and didn't pay attention to other aspects of localization. In response to this reasoning, we propose the new model changes to accommodate localization capabilities beyond the language and encoding schemes support. Localization is a broad concept and is not limited to supporting different languages (for more solid background, refer to section 5. Software Globalization/ Localization).

<u>In addition, there is another issue with usability</u>, since the original class included two string arrays only, one for input locales and the other for output locales, which isn't useful to the service intended by introduction of this class, and doesn't add valuable information to serve the needs of configuration of localization capabilities of different elements in the model, since these two array attributes could be simply added to the base class (Capabilities or ManagedElement) without affecting the semantics, and separation in a standalone class is a kind of specialization that could be better exploited by adding more attributes and functionality to this class.

By incorporating the proposed changes the support of configuration of localization capabilities is more mature and useful in serving the needs of model elements to deal with localization aspects of each other and with external systems. This goes on the same line with the above quoted statement from "CIM Diagnostics Model White Paper" after initial introduction of the original class, saying "*The change was written as generically as possible, specifically supporting diagnostics with the intent that it be generalized for broader use in the future."*

6.5. Related Work

This section reviews the research works have been done so far for autonomic computing (not merely under this name) in both academia and industry sectors, as stated in "Autonomic Computing : Emerging Trends and Open Problems" [28].

As autonomic computing is a multi-disciplinary research area, a remarkable number of projects in computer science, software engineering, artificial intelligence are related to it. Because of space constraint, we are only able to address a part of these projects in this survey paper (mostly supported by IBM) [28]:

- OceanStore, UC Berkeley : designs a global persistent data store for scaling to billions of users.
- AntHill, University of Bologna : supports the design, implementation and evaluation of P2P applications based on multi-agent and evolutionary programming borrowed from complex adaptive systems (CAS).
- Recovery-Oriented Computing, UC Berkeley/Stanford : investigates novel techniques for building highly-dependable Internet services, recovery from failures rather than failure-avoidance.
- Autonomia, University of Arizona : provides the application developers with all the tools required to specify the appropriate control and management schemes to maintain any quality of service requirements.
- eBiquity, University of Baltimore County : explores the interactions between mobile, pervasive computing, multi-agent systems and artificial intelligence techniques.
- Software Rejuvenation, Duke University : develops fault management techniques aimed at cleaning up the system internal state to prevent the occurrence of more severe crash failures in the future.

In additions to the above mentioned researches that focused mainly on developing autonomic systems, there are also some researches in academia focused their work on standards supporting autonomic computing as related to DMTF effort in this regard. The following are examples of such works :

- "Automating Applications Management in the Enterprise using DMTF Information Models" [29] : The initial steps to provide self managing application environments are now being taken – a paradigm known as "autonomic computing" is in it's infancy of evolution. DMTF specifications that are currently being developed for management of distributed applications form the basis for some of this work. There have been numerous proposed models of how one achieves self management. This paper formulates the research problems and basis for "lights out" management of enterprise application environments. It also illustrates the researchers' approach with a way of managing simple web applications based on Java Servlets.
- "Integrating CIM/WBEM with the Java Enterprise Model" [30] : This paper discusses the experiences of developing a multi-tiered Heterogeneous Alarm Management System based on the Java2 Enterprise Edition, with CIM/WBEM as the interface to the legacy and mixed protocol architectures. It also discusses the simulation of a SNMP/CMIP based heterogeneous network environment.
- "CAMELEON: A CIM "Modelware" platform for distributed integrated management" [31] : CAMELEON is a distributed management platform for complex systems. This project provides an innovative approach for global and

distributed management system involving object-oriented technologies such as CIM, CORBA and Java.

- “Using CIM to Realize Policy Validation within the Ponder Framework” [32] : This paper discusses how CIM can be used within the Ponder Policy Framework to validate network policies that apply to a Differentiated Services (DiffServ) domain against the capabilities of the individual network elements that comprise the DiffServ domain.
- “A Network Management Platform adaptable to CIM Model evolution” [33] : The proposed platform framework is based on the CIM information model and its PCIM extension. The framework allows to extend the platform capability and enables an automatic validation while eliminating the need to write adaptation code. The researchers present the platform framework in the case of the policy manager. They illustrate how the platform can easily evolve to incorporate the new classes corresponding to the new services using the example of the Virtual Private Network (VPN) policy management.
- "Policy-Maker", a Toolkit for Policy-Based Security Management [36] : “Policy-Maker” is an implementation of the researchers’ concept for the security management of heterogeneous networks. It is entirely based on the Common Information Model (CIM) and the Web-Based Enterprise Management (WBEM) architecture, which is an industry standard of the Distributed Management Task Force (DMTF). Furthermore, CIM-based policy models for some concrete security mechanisms (e.g. IP-Firewalls) have been designed and implemented for testing the “Policy-Maker”. At the beginning of the concept development phase there existed a CIM model for IPSec policies. The researchers developed CIM models for some further security mechanisms, e.g. IP and CORBA firewalls and IDS . They presented their models to the DMTF in January 2003. The CIM models of firewalls were the basis for implementing and testing our concepts.
- “Distributed Network Security” [37] : The approach discussed in this article splits into three parts – first the researchers invent distributed sensors which enlarge the amount of data available for analysis by accessing information directly at its source. To integrate these into the classic border oriented system they create an abstract interface and management system, based on the Common Information Model. Finally they will divide the management system itself into independent components, distribute them over the network and gain significant increase of performance. Most of the changes invented in this project were discussed during the weekly phone conference of the SPAM working group in DMTF and have been improved with the feedback of the group. Currently they are reworking the first partial implementations and try to create a working prototype of the complete system. It is planned to discuss the model within the working group again and bring in the results into the standardization process.

7. Validation of the Proposed Model

7.1. Validation Methodology

Basically, the process followed by DMTF for commitment on changes and modification of any of its developed standards depends on discussions between working groups and steering committee resulting decisions to accept or reject changes of each standard release. However, this approach can't be used here for an externally proposed work – not internally generated from DMTF groups discussions. That raises the need for a validation method to show the correctness and rationale of the proposed work.

"The ultimate goal of the verification and validation process is to establish confidence that the software system is 'fit for purpose'. This doesn't mean that the program must be completely free of defects. Rather, it means that the system must be good enough for its intended use. The level of required confidence depends on the system's purpose, the expectations of the system users and the current marketing environment for the system." [16]

"Software inspections don't require the program to be executed so may be used as a verification technique before programs are implemented. During an inspection, you examine the source representation of a system. This could be a system model, a specification or high level language code … This doesn't mean that inspections should completely replace system testing. Rather, they should be used as an initial verification process." [16]

Traditionally, the configuration design of IT service and infrastructure management environments has been a process largely governed by heuristics. In the last few years it became evident that with the business of organizations relying more and more on their IT infrastructure there is a need for the verification and validation of the nonfunctional properties. These service level and information security requirements have to be satisfied in large, heterogeneous IT systems. However, heuristic design of IT management configurations cannot cope with complex systems. Employment of the Model Driven Architecture (MDA) approach, well-established by now, to model, analyze and plan IT infrastructure management configurations can be advantageous. [34]

7.2. Model Driven Architecture (MDA)

The MDA is a new way of developing applications and writing specifications, based on a platform-independent model (PIM) of the application or specification's business functionality and behavior. A complete MDA specification consists of a definitive platform-independent base model, plus one or more platform-specific models (PSM) and sets of interface definitions, each describing how the base model is implemented on a different middleware platform. A complete MDA application consists of a definitive PIM, plus one or more PSMs and complete implementations, one on each platform that the application developer decides to support.

Software technology is transient. When a standard is irrevocably bound to a particular technology, the quality work and experience that go into its conceptual design disappear along with the technology. The MDA approach makes it possible to save the conceptual design, which is the most valuable part of the investment. But even if the relevant technologies are in no imminent danger of disappearing, the MDA helps to avoid duplication of effort and other needless waste.

7.3. How will Standards take advantage of the MDA?

In order to benefit an industry, a standard must be used by a critical mass of companies. Technology-specific standards will have trouble getting established where platform incompatibility prevents achieving this critical mass. Sometimes the problem is even deeper than this: In some industries, architecturally excellent standards have been adopted in the formal sense but failed to gain hold because they were written for a platform that few companies were willing to support.

MDA completely removes these roadblocks. Under MDA, the functional description of every standard is technology independent, and the architecture is capable of producing interoperating implementations on multiple platforms. This allows an industry to define the business functionality and behavior of its standards as a PIM, and then produce PSMs and implementations on whatever platforms the participants require.

In addition, technology-based standards become obsolete as their base platform ages; in fact they lose some of their appeal as soon as a new platform gets some play in the industry press. This is not a problem for MDA-based specifications: Because they are based on platform-independent PIMs and can be made to interoperate with new platforms, or even re-implemented on them, through the MDA development process, MDA-based applications and specifications are truly "future-proof".

Bottom line: Using MDA, the industry gets a standard, every company uses it, none are forced to switch platforms in order to benefit, and the standard lasts as long as the industry process that it serves. Everybody wins! [27]

7.4. An MDA perspective on proposed work

While the MDA originated in the OMG, there is nothing to prevent other standards organizations from benefiting from it. The primacy of the conceptual design makes it easier for previous work to be adopted in OMG and for OMG work to be realized in the contexts of other standards organizations. Moreover, by factoring out the aspects of standards that are peculiar to particular technologies, the MDA eliminates many of the bad reasons that are often used to justify "yet another" standard. One may well need to build "yet another" PSM, or perhaps many of them, but the more difficult and expensive process of conceptual design need not be repeated. Unlike so many transient technologies, the MDA represents a meaningful step towards reducing the cost of achieving and maintaining "interoperability." [35]

Relating MDA’s objective of interoperability to the same main objective of DMTF’s standards, among which is CIM, being also interoperability, we can measure the correctness and validation of the proposed model changes based on MDA metrics and concepts.

The CIM model is designed to be implementation-independent. It is independent of the method used for implementation, whether it is used in database modeling, in objects design in OOP environment, or in information exchange/parameter passing using a neutral data exchange method like XML representation. (Refer to section 4.CIM – Common Information Model, for more details). Thus, the CIM model can be regarded as a PIM according to MDA concepts.

The core model of an MDA standard specifies only its business functionality and behavior, divorced from platform-specific aspects. Working in this environment, we can focus on business detail exclusively, working and reworking this aspect of the application until it gets exactly right. And, should an aspect of an implementation not work correctly in an early implementation, it is easy to see if this is due to a failure to model the business behavior correctly, or a fault of the platform-specific code. [15]

<u>Starting from this point, the correctness of the proposed model can be validated by clarifying correspondence between the proposed model components and the modeled business requirements. This can be shown as detailed below :</u>

Business Requirement	Model Representation

Locale is represented in terms of language-territory (country) pairs	Attributes :
	`Locale.LocaleName`
	`LocaleSettingData.Country`
	`LocaleSettingData.Language`
	
	Associations :
	`CIM_SupportedLocale`
	`CIM_LocaleSetting`

Locale is used as a mechanism to switch language or cultural dependent features, like calendar, time, monetary, formatting, and specific translations of key sentences.	Attributes :
	`LocaleSettingData.Script`
	`LocaleSettingData.bidiSupport`
	`LocaleSettingData.AnnotationSupport`
	`LocaleSettingData.VowelizationSupport`
	`LocaleSettingData.CurrencyName`
	`LocaleSettingData.CurrencySymbol`
	`LocaleSettingData.DateFormates`
	`LocaleSettingData.FlowDirections`
	`LocaleSettingData.PunctuationSymbols`
	`LocaleSettingData.CourtesyTitles`
	`LocaleSettingData.Greetings`
	`LocaleSettingData.MeasureUnits`
	`LocaleSettingData.Vowels`
	`LocaleSettingData.CharTransliteration`
	Along with their values & value maps in the `Locale` class.
	
	In addition, all attributes of `TimeZoneSettingData` class are relevant to this business requirement, they will not be listed here since no changes were proposed for this class.
	
	Associations :
	`CIM_LocaleSetting`
	`CIM_LocaleTime`

Code Page support is one of localization matters. It's necessary for handling characters representation from different scripts and languages.	Attributes :
	`EncodingScheme.SchemeStr`
	`EncodingScheme.MappingTable`
	
	Methods :
	`EncodingScheme.EncodeChar`
	`      ([IN]charToEncode:char16)`
	`EncodingScheme.DecodeChar`
	`      ([IN,Octetstring]codestring:uint8[])`
	
	Associations :
	`CIM_SupportedSchemes`

Table 6. Correspondence between business requirements and proposed model components

Moreover, the next illustrated sequence diagram (section 7.5) provides interactive description of the applicability scenarios of the model classes, emphasizing the satisfaction of business requirements in action.

7.5. Sequence Diagram Illustration of Proposed Model Usage Scenarios

This sequence diagram shows the scenario followed by an external element (system) in order to interact with a certain managed element regarding exploration of its localization capabilities and supported operations.

The external system first pings the managed element to check its ability to handle localization issues, check support of a specific locale and a specific encoding scheme by name, and gets the locale object and the encoding scheme object. Then the external system continues working with several localization operations through the Locales and Encoding Schemes supported for this managed element, and their related attributes. It gets the time offset and the locale expression for the ‘welcome’ greeting from the locale object, as well as the character encoding and decoding from the encoding scheme object.

All other use cases can be handled similar to the example case described below.

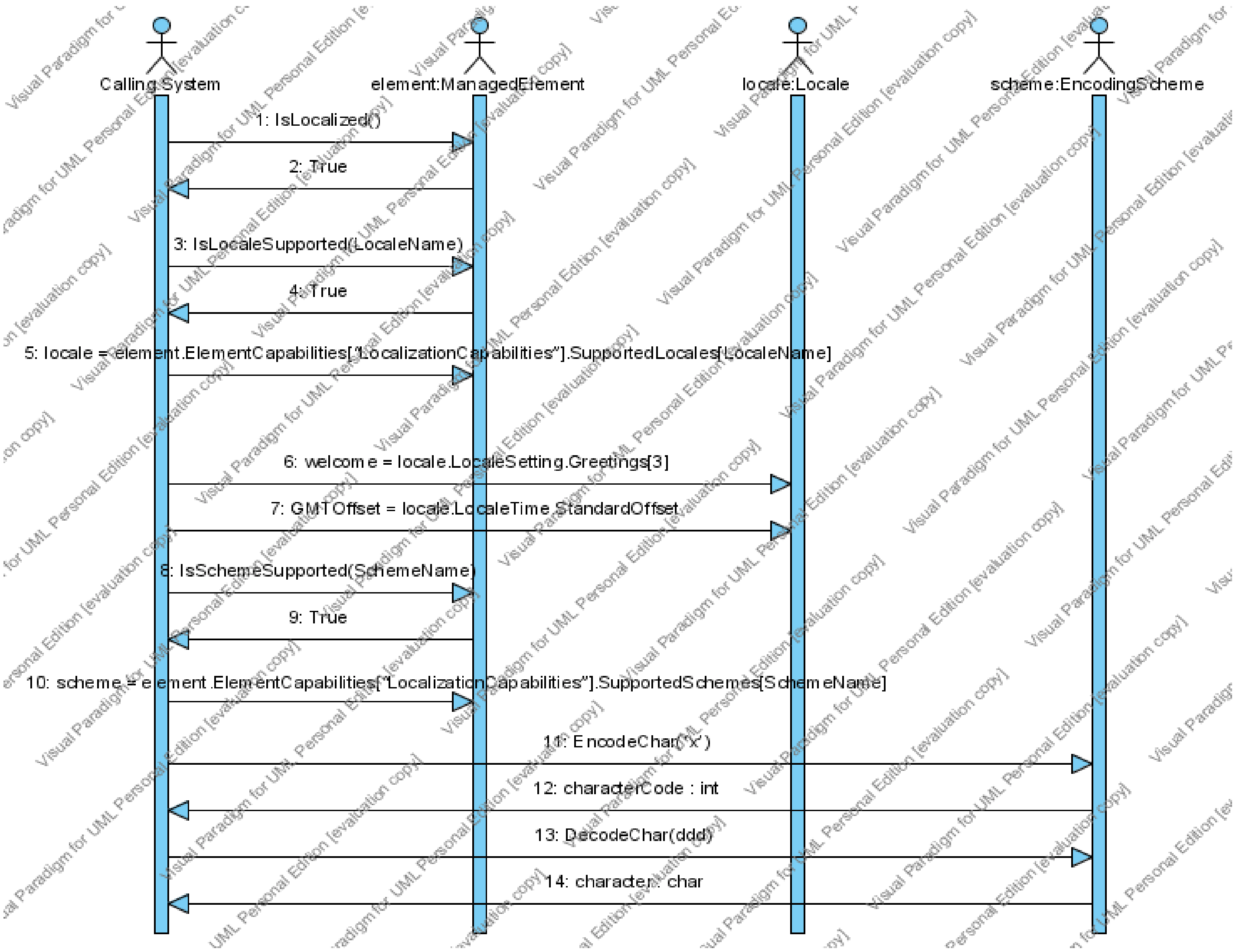

Calling:System
element:ManagedElement
locale:Locale
scheme:EncodingScheme
1: IsLocalized()
2: True
3: IsLocaleSupported(LocaleName)
4: True
5: locale = element.ElementCapabilities["LocalizationCapabilities"].SupportedLocales[LocaleName]
6: welcome = locale.LocaleSetting.Greetings[3]
7: GMTOffset = locale.LocaleTime.StandardOffset
8: IsSchemeSupported(SchemeName)
9: True
10: scheme = element.ElementCapabilities["LocalizationCapabilities"].SupportedSchemes[SchemeName]
11: EncodeChar('x')
12: characterCode : int
13: DecodeChar(ddd)
14: character : char

Figure 11. Example Use Case Sequence Diagram

7.6. Pseudo Code of implementation prototype

The following is a Java-like pseudo code of a prototype implementation of the proposed model. It illustrates implementation steps and interaction access methods between the related classes.

```
class ManagedElement {
      string Caption;
      string Description;
      string ElementName;
      Capabilities[] ElementCapabilities;
}
```

```
abstract class Capabilities extends ManagedElement {
      string InstanceID;
      string ElementName;
}
```

```
class LocalizationCapabilities extends Capabilities {
      boolean IsLocalized;
      Locale[] SupportedLocales;
      EncodingScheme[] SupportedSchemes;

      public LocalizationCapabilities(string elementName) {
            this.ElementName = elementName;
      }

      boolean IsLocaleSupported(string locale){
            for(int i=0; i<SupportedLocales.size();i++) {
                  if(SupportedLocales[i].LocaleName.equals(locale))
                        return true;
            }
            return false;
      }
      boolean IsSchemeSupported(string scheme) {
            for(int i=0; i<SupportedSchemes.size();i++) {
                  if(SupportedSchemes[i].SchemeName.equals(scheme))
                        return true;
            }
            return false;
      }
}
```

```
class Locale extends ManagedElement {
      static string[] FlowDirectionsOrder =
                        [“rtl”, “ltr”, “tb-rl”, “lr-tb”, “rl-tb”];
      static string[] CourtesyTitlesOrder =
                        [“Ms.”, “Mrs.”, “Mr.”, “Sir”, “Dr.”, “Prof.”];
      static string[] GreetingsOrder =
                        [“G.M.”, “GAN”, “G.N.”, “Welcome”, “Bye”];
      static string[] MeasureUnitsOrder =
                        [“Weight”, “Length”, “Freq.”, “Temp.”];
      static char[] PunctuationSymbolOrder =
                        [‘,’, ‘.’, ‘;’, ‘/’, ‘\’, ‘:’, ‘“’, ‘‘’];
      static string[] VowelsOrder =
                        [“a”, “i”, “u”, “e”, “o”, “aa”, “ii”, “uu”,
“ee”, “oo”, “stress”, “consonant”];

//Define the standard repository used for setting values for the fields
//of LocaleSettingData and TimeZoneSettingData objects that will
//reference a specific Locale object. In this case, it’s CLDR data
//repository
      static string LocaleIdentifier = ‘CLDR’;


      string LocaleName;
      LocaleSettingData LocaleSetting;
      TimeZoneSettingData LocaleTime;

      public Locale(string localeName) {
            this.LocaleName = localeName;
      }
}
```

```
abstract class SettingData extends ManagedElement {
      string InstanceID;
      string ElementName;
}
```

```
class LocaleSettingData extends SettingData {
//Fields of this class will be set either by loading their values from
//the XML data repository files of the repository defined by the
//LocaleIdentifier field in the Locale class, or by setting their
//values individually like in using resource bundles or hard coded
//values.
      string Country;
      string Language;
      string Script;
      boolean bidiSupport;
      boolean AnnotationSupport;
      boolean VowelizationSupport;
      string CurrencyName;
      char CurrencySymbol;
      string[] DateFormates;
      boolean[] FlowDirections;
      char[] PunctuationSymbols;
      string[] CourtesyTitles;
      string[] Greetings;
      string[] MeasureUnits;
      char[] Vowels;
      string[] CharTransliteration;

      public LocaleSettingData(string elementName) {
            this.ElementName = elementName;
      }
}
```

```
class TimeZoneSettingData extends SettingData {
//Fields of this class will be set either by loading their values from
//the XML data repository files of the repository defined by the
//LocaleIdentifier field in the Locale class, or by setting their
//values individually like in using resource bundles or hard coded
//values.
      string StandardName;
      string StandardCaption;
      sint StandardOffset;
      uint StandardMonth;
      sint StandardDay;
      sint StandardDayOfWeek;
      Date StandardStartInterval;
      string DaylightName;
      string DaylightCaption;
      sint DaylightOffset;
      uint DaylightMonth;
      sint DaylightDay;
      sint DaylightDayOfWeek;
      Date DaylightStartInterval;

      public TimeZoneSettingData (string elementName) {
            this.ElementName = elementName;
      }
}
```

```
abstract class EncodingScheme extends ManagedElement {
      string SchemeName;
      uint[][] MappingTable;

      unit[] EncodeChar(char charToEncode);
      char DecodeChar(unit[] codestring);
}
```

```
class UnicodeEncodingScheme extends EncodingScheme {
//No need for the MappingTable in case of usage of library
//methods for encoding/decoding. It’s more useful for custom
//encoding/decoding schemes.

      public UnicodeEncodingScheme (string schemeName) {
            this.SchemeName = schemeName;
      }

      unit[] EncodeChar(char charToEncode) {
            //Use platform specific library methods, like java, to
            //encode character
      }

      char DecodeChar(unit[] codestring) {
            // Use platform specific library methods, like java, to
            //decode character
      }
}
```

```
class TestManagedElement extends ManagedElement {
//Constructor Method
      public TestManagedElement(string elementName) {
            this.Caption =
                  “A test subclass of the CIM ManagedElement class”;
            this.Description = “This class is created for test purposes
of the proposed model modifications for the LocalizationCapabilities
class. The implementation is simple and intended to be demonstrative of
the example configuration steps.”
            This.ElementName = elementName;
      }

//A public method to configure the localization settings of the
//ManagedElement object
      public void configureManagedElement(void){

//Extract the LocalizationCapabilities object associated with this
//ManagedElement object from the ElementCapabilities association which
//relates every ManagedElement object with its Capabilities objects,
//among which is the LocalizationCapabilities object. Use the object
//key to access it.
      LocalizationCapabilities elementLocalizationCapabilities =
            (this.ElementCapabilities.add
      (new LocalizationCapabilities(“LocalizationCapabilities”))
            );

//Use the above extracted LocalizationCapabilities object to configure
//this ManagedElement object localization settings.
            elementLocalizationCapabilities.isLocalized = true;

//Create and configure the Locale objects that should be supported by
//the LocalizationCapabilities object of the ManagedElement object in
//question.
            Locale Arabic_Egypt = new Locale(“AR_EG);
            ConfigureArabic_EgyptSettings(Arabic_Egypt);
            ConfigureArabic_EgyptTime(Arabic_Egypt);
            Locale Japanese_Japan = new Locale(“JP_JP);
            ConfigureJapanese_JapanSettings(Japanese_Japan);
            ConfigureJapanese_JapanTime(Japanese_Japan);

            elementLocalizationCapabilities.SupportedLocales.add
                  (Arabic_Egypt);
            elementLocalizationCapabilities.SupportedLocales.add
                  (Japanese_Japan);
            elementLocalizationCapabilities.SupportedSchemes.add
                  (new UnicodeEncodingScheme(“Unicode”));
      }

      private void ConfigureArabic_EgyptSettings(Locale Arabic_Egypt) {
            Arabic_Egypt_Setting = (LocaleSettingData)
                  (Arabic_Egypt.LocaleSetting =
                        new LocaleSettingData(“AR_EG_Setting”);
                  );
//The approach used here for setting values for the LocaleSettingData
//fields is to hard code the values, for illustration purposes. Other
//options to set their values include loading their values from the
//corresponding XML files in the CLDR locale data repository, or
//loading their values from a customized resource bundle files.
            Arabic_Egypt_Setting.Country = “Egypt”;
            Arabic_Egypt_Setting.Language = “Arabic”;
            Arabic_Egypt_Setting.Script = “Arabic”;
            Arabic_Egypt_Setting.bidiSupport = true;
            Arabic_Egypt_Setting.AnnotationSupport = false;
            Arabic_Egypt_Setting.VowelizationSupport = true;
            Arabic_Egypt_Setting.CurrencyName = “EGP”;
            Arabic_Egypt_Setting.CurrencySymbol = “”;
            Arabic_Egypt_Setting.DateFormates =
                  [“DD/MM/YYYY”,”DD/MMM/YYYY”, “DD/MM/YY”];
            Arabic_Egypt_Setting. FlowDirections =
                  [true, false, false, false, true];
            Arabic_Egypt_Setting.PunctuationSymbols =
                  [‘،’, ‘.’, ‘؛’, ‘/’, ‘\’, ‘:’, ‘”’, ‘’’’];
            Arabic_Egypt_Setting.CourtesyTitles =
                  ["آنسة", "سيدة", "سيد", "سيد", "دكتور", "أستاذ"];
            Arabic_Egypt_Setting.Greetings =
            ["صباح الخير", "مساء الخير", "مساء الخير", "مرحباً", "مع السلامة"];
            Arabic_Egypt_Setting.MeasureUnits =
                  ["كيلو", "متر", "هرتز", "درجة مئوية"];
            Arabic_Egypt_Setting.Vowels =
            [’ـَ’, ’ـِ’, ’’, ’’, ’ـُ’, ’ا’, ’ي’, ’’, ’’, ’و’, ’ـّ’, ’ـْ’];
            Arabic_Egypt_Setting.CharTransliteration =
[“a”,”b”,”t”,”th”,”j”,”h”,”kh”,”d”,”th”,”r”,”z”,”s”,”sh”,”s”,”d”,”t”,”t
h”,”e”,”gh”,”f”,”k”,”k”,”l”,”m”,”n”,”h”,”w”,”y”];
      }

      private void ConfigureJapanese_JapanSettings(Locale
Japanese_Japan) {
            Japanese_Japan_Setting = (LocaleSettingData)
                  (Japanese_Japan.LocaleSetting =
                        new LocaleSettingData(“JP_JP_Setting”);
                  );
//The approach used here for setting values for the LocaleSettingData
//fields is to hard code the values, for illustration purposes. Other
//options to set their values include loading their values from the
//corresponding XML files in the CLDR locale data repository, or
//loading their values from a customized resource bundle files.
            Japanese_Japan_Setting.Country = “Japan”;
            Japanese_Japan_Setting.Language = “Japanese”;
            Japanese_Japan_Setting.Script = “Japanese”;
            Japanese_Japan_Setting.bidiSupport = false;
            Japanese_Japan_Setting.AnnotationSupport = true;
            Japanese_Japan_Setting.VowelizationSupport = false;
            Japanese_Japan_Setting.CurrencyName = “YEN”;
            Japanese_Japan_Setting.CurrencySymbol = “¥”;
            Japanese_Japan_Setting.DateFormates =
                        [“DD 日 MM 月 YYYY 年”];
            Japanese_Japan_Setting. FlowDirections =
                        [false, true, true, true, false];
            Japanese_Japan_Setting.PunctuationSymbols =
                        [‘、’, ‘。’, ‘’, ‘/’, ‘\’, ‘:’, ‘「’, ‘「’];
            Japanese_Japan_Setting.CourtesyTitles =
                        ["さん", "さん", "さん", "様", "先生", "先生"];
            Japanese_Japan_Setting.Greetings =
            ["おはよう", "こんばんわ", "こんばんわ", "ようこそ", "さようなら"];
            Japanese_Japan_Setting.MeasureUnits =
                        ["キロ", "メートル", "ヘルツ", "度"];
            Japanese_Japan_Setting.Vowels =
            [’’, ’’, ’’, ’’, ’’, ’’, ’’, ’’, ’’, ’’, ’’, ’’];
            Japanese_Japan_Setting.CharTransliteration =
[“a”,”i”,”u”,”e”,”o”,”ka”,”ki”,”ku”,”ke”,”ko”,”sa”,”shi”,”su”,”se”,”so”
,”ta”,”chi”,”tsu”,”te”,”to”,”na”,”ni”,”nu”,”ne”,”no”,”ha”,”hi”,”fu”,”he
”,”ho”,”ma”,”mi”,”mu”,”me”,”mo,”ya”,”yu”,”yo”,”ra”,”ri”,”ru”,”re”,”ro”,
”wa”,”n”];
      }

      private void ConfigureArabic_EgyptTime(Locale Arabic_Egypt) {
            Arabic_Egypt_Time = (TimeZoneSettingData)
                  (Arabic_Egypt.LocaleTime =
                        new TimeZoneSettingData(“AR_EG_Time”);
                  );
//The approach used here for setting values for the TimeZoneSettingData
//fields is to hard code the values, for illustration purposes. Other
//options to set their values include loading their values from the
//corresponding XML files in the CLDR locale data repository, or
//loading their values from a customized resource bundle files.
            Arabic_Egypt_Time.StandardName = “Egypt Time”;
            Arabic_Egypt_Time.StandardCaption = “EET”;
            Arabic_Egypt_Time.StandardOffset = 2;
            Arabic_Egypt_Time.StandardMonth = 10; //Sep.
            Arabic_Egypt_Time.StandardDay = 28;
            Arabic_Egypt_Time.StandardDayOfWeek = -5;
            Arabic_Egypt_Time.StandardStartInterval =
                                                new Date(“0.0:0”);
            Arabic_Egypt_Time.DaylightName = “Egypt Time”;
            Arabic_Egypt_Time.DaylightCaption = “EEST”;
            Arabic_Egypt_Time.DaylightOffset = 3;
            Arabic_Egypt_Time.DaylightMonth = 5; //Apr.
            Arabic_Egypt_Time.DaylightDay = 21;
            Arabic_Egypt_Time.DaylightDayOfWeek = -5;
            Arabic_Egypt_Time.DaylightStartInterval =
                                                new Date(“0.0:0”);
      }

      private void ConfigureJapanese_JapanTime(Locale Japanese_Japan) {
            Japanese_Japan_Time = (TimeZoneSettingData)
                  (Japanese_Japan.LocaleTime =
                        new TimeZoneSettingData(“JP_JP_Time”);
                  );
//The approach used here for setting values for the TimeZoneSettingData
//fields is to hard code the values, for illustration purposes. Other
//options to set their values include loading their values from the
//corresponding XML files in the CLDR locale data repository, or
//loading their values from a customized resource bundle files.
            Japanese_Japan_Time.StandardName = “Japan Standard Time”;
            Japanese_Japan_Time.StandardCaption = “JST”;
            Japanese_Japan_Time.StandardOffset = 9;
            Japanese_Japan_Time.StandardMonth = 0;
            Japanese_Japan_Time.StandardDay = 0;
            Japanese_Japan_Time.StandardDayOfWeek = 0;
            Japanese_Japan_Time.StandardStartInterval =
                                                new Date(“0.0:0”);
            Japanese_Japan_Time.DaylightName = “Japan Standard Time”;
            Japanese_Japan_Time.DaylightCaption = “JST”;
            Japanese_Japan_Time.DaylightOffset = 9;
            Japanese_Japan_Time.DaylightMonth = 0;
            Japanese_Japan_Time.DaylightDay = 0;
            Japanese_Japan_Time.DaylightDayOfWeek = 0;
            Japanese_Japan_Time.DaylightStartInterval =
                                                    new Date(“0.0:0”);
      }
}
```

```
class ExternalSystem {

      TestManagedElement element=new TestManagedElement(“TestElement”);

      void main(void) {
            element.configureManagedElement();

            if(element.isLocalized()){
                  if(element.IsLocaleSupported(“AR_EG”)) {
                        Locale locale =
element.ElementCapabilities[“LocalizationCapabilities”].SupportedLocale
s[“AR_EG”];
                        System.output.println(“Welcome in AR_EG is :
“ + locale.LocaleSetting.Greetings[3]);
                        System.output.println(“GMT offset in AR_EG is :
“ + locale.LocaleTime.StandardOffset);
                  }
                  else System.output.println(“Arabic is not
supported”);

                  if(element.IsSchemeSupported(“Unicode”)) {
                        EncodingScheme scheme =
element.ElementCapabilities[“LocalizationCapabilities”].SupportedScheme
s[“Unicode”];
                        System.output.println(“The code of the first
Arabic letter in Unicode is : “ + scheme.EncodeChar(‘أ’));
                        System.output.println(“The character
corresponding to code value 0x0686 in Unicode is : “ +
scheme.DecodeChar(0x0669));
                  }
                  else System.output.println(“Unicode is not
supported”);
            }
            else
                  System.output.println(“The  Managed  Element  is  not
localizable. Exiting.”);
      }
}
```

The expected output of running the main method of the ExternalSystem class in this prototype is to print the following lines to the standard output :

```
Welcome in AR_EG is : مرحباً
GMT offset in AR_EG is : 2
The code of the first Arabic letter in Unicode is : 0x0623
The character corresponding to code value 0x0669 in Unicode is : ٩
```

8. Conclusion & Future Work

Conclusion

In this work, we proposed model modifications to the LocalizationCapabilities class in the core model of CIM standard, which is released by DMTF organization. We clarified the significance of this modification, and set forth validation of model correctness by correspondence to business field requirements. A prototype implementation was presented as well to show the integrity of classes' interactions and achievement of the stated business goal of supporting configuration of localization attributes in the system management domain.

Although implementation of standards is still a little behind in practice, acceptance and adoption of interoperation standards by companies in industry is a roadmap for effectiveness and success of standardization efforts. Experimental industrial applications of proposed standard modifications dictate approval or rejection of the proposed work, as implied by the processing methodology of DMTF organization.

In autonomic computing research, academic and industrial projects realized remarkable parts of the autonomic computing vision. However, there are still open problems to deal with in this promising research area. [28]

This work will be presented to DMTF for further assessment and proposal for inclusion in the organization's standardization efforts in CIM.

Future Work

As the business requirements and software technologies keep changing and evolving all the way, standards will keep coping with and driving these changes. It's determinate that standards are required to play a more effective role in business development and market place. Continuous investigation and enhancement of the defined standards is inevitable. As new versions of CIM standard come out, revision and revisiting of this work is important. Areas of possible enhancement of the proposed work may include :

- Supporting more localization features; cultural, political, lingual and geographical.
- Modeling the locale data repository (like CLDR) in a more general and manageable manner, like representing it directly in the model by creating specific classes/relations to model it rather than specifying it as a single string field.
- More features of encoding schemes can be exploited, such as character formatting in several format methods.
- Some kind of relation or dependency can be created between the encoding scheme and the locale settings, like coding characters according to known preference in a specific locale. For example, use of specific font or character set in certain locale. This is still achievable using the proposed model but with more concentration on implementation effort, while some future work may leverage this effort to the model itself.
- It may be feasible to add some level of dynamicity to the model to allow locale settings to adapt with the current temporal political and social status.

References

Publications

1- On demand Operating Environment Creating Business Flexibility (Red Book)
2- An Architectural Blueprint for Autonomic Computing [http://www-03.ibm.com/autonomic/pdfs/ACBP2_2004-10-04.pdf]
3- The Autonomic Computing Edge - The Standard way of autonomic computing [http://www-128.ibm.com/developerworks/library/ac-edge2/]
4- Introduction to Autonomic Computing (developer works)
5- A Practical Guide to the IBM Autonomic Computing Toolkit (Red Book)
6- The Autonomic Computing Edge - Autonomic computing heats up in Japan [http://www.ibm.com/developerworks/library/ac-edge/]
7- CIM Concepts White Paper - DSP0110.pdf (DMTF specification)
8- CIM Core Model White Paper - DSP0111.pdf (DMTF specification)
9- Management application programming, Part 2 Introduction to WBEM and the CIM [http://www.ibm.com/developerworks/java/library/j-wbem/]
10-CIM Infrastructure Specification - DSP0004V2[1].3_final.pdf (DMTF specification)
11-CIM_Core.pdf (DMTF specification)
12-e-business Globalization - Solution Design Guide (Red Book)
13-diagnostics_whitepaper_v0_9.51.pdf (DMTF specification)
14-Specification for CIM Operations over HTTP – DSP0200.pdf (DMTF specification)
15-Developing in OMG's Model-Driven Architecture - Jon Siegel and the OMG Staff Strategy Group (OMG White Paper)
16-Ian Sommerville, "Software Engineering", 6th Ed., Addison-Wesley.

Websites

17-http://www.ibm.com/autonomic
18-http://www.dmtf.org
19-http://www.ibm.com/developerworks
20-http://www.ibm.com/redbooks
21-http://www.xencraft.com
22-http://www.i18nguy.com
23-http://www.unicode.org
24-http://www.unicode.org/cldr/
25-http://www.intel.com
26-http://www.lisa.org
27-http://www.omg.org/mda

Academic Papers

28-Autonomic Computing : Emerging Trends and Open Problems - Mazeiar Salehie & Ladan Tahvildari, Dept. of Elect. and Comp. Eng., University of Waterloo. DEAS 2005, May 21, 2005, St. Louis, Missouri, USA.
29-Automating Applications Management in the Enterprise using DMTF Information Models - Umesh Bellur, Indian Institute of Tech. Bombay, Powai, Mumbai
30-Integrating CIM/WBEM with the Java Enterprise Model - Kenneth Carey & Fergus O' Reilly, Adaptive Wireless Systems Group, Department of Electronic Engineering, Cork Institute of Technology, Cork, IRELAND
31-CAMELEON: A CIM "Modelware" platform for distributed integrated management - M. Sibilla, A. Barros de Sales, Y.Steff, T. Desprats, D. Marquié,

Institut de Recherche en information de Toulouse (IRIT) & François Jocteur - Monrozier, C. Lasserre, CNES & Anne-Isabe lle Rivière, Alcatel CIT

32-Using CIM to Realize Policy Validation within the Ponder Framework - Leonidas Lymberopoulos, Emil Lupu and Morris Sloman, Imperial College, Department of Computing, 180 Queen's Gate, SW7 2BZ, London, UK

33-A Network Management Platform adaptable to CIM Model evolution - Nathalie Rico, Omar Cherkaoui and Elmi Hassan, Univerisité de Montréal, University of Quebec in Montreal

34-Model-Based Design of System Management - Fault Tolerant Systems Research Group, Department of Measurement and Information Systems, Budapest University of Technology and Economics

35-Impact of Model-Driven Standards - David Flater, National Institute of Standards and Technology, 100 Bureau Drive, Stop 8260 Gaithersburg, MD 20899-8260, U.S.A.

36-"Policy-Maker", a Toolkit for Policy-Based Security Management - Andreas Pilz, Institute for Data Processing, TU München

37-Distributed Network Security - Dipl.-Ing. Oliver Welter, Dipl.-Ing. Andreas Pilz, Technische Universitaet Muenchen

Appendices

Appendix A

“Internationalization Considerations” section in DSP0200

4.8. Internationalization Considerations

This section defines the capabilities of the CIM HTTP Mapping with respect to IETF policy guidelines on character sets and languages [19].

In this specification, human-readable fields can be found within a response or request entity body. In all cases, any human-readable content is encoded using XML (which has explicit provisions for character set tagging and encoding) and requires that XML processors read XML elements encoded, at minimum, using the UTF-8 [15] encoding of the ISO 10646 multilingual plane.

Properties that are not of type string or string array MUST NOT be localized.

Since keys are only writeable on instantiation, key values MUST NOT be localized. See the CIM Specification, v 2.2 for further details.

XML examples in this specification demonstrate use of the charset parameter of the Content-Type header, as defined in [10], as well as the XML attribute on the <?xml> processing instruction, which together provide charset identification information for MIME and XML processors. This specification mandates that conforming applications MUST support at least the "UTF-8" charset encoding [19] in the Content-Type header, and the "UTF-8" value for the XML encoding attribute.

XML also provides a language tagging capability for specifying the language of the contents of a particular XML element, based on use of IANA registered language tags [20] in combination with ISO 639-1 in the xml:lang attribute of an XML element to identify the language of its content and attributes. Section 3.10 of RFC 2616 defines how the two character language code, ISO 639-1 is used as the primary-tag. The language-tag MUST be registered by IANA.

The XML CIM DTD [2,11] declares this attribute on any of the XML elements, and therefore conforming applications SHOULD use this attribute when specifying the language a particular element is encoded in, string and string array attributes and qualifiers. See the usage rules on this element defines by the World Wide Web Consortium, XML 1.0, second edition. The attribute MAY be scoped by the instance or a class and SHOULD NOT by a property because instances or classes SHOULD be localized in a one language.

This specification defines a number of names of HTTP headers and their values. These are constructed using standard encoding practices so as to always have an HTTP-safe ASCII representation. Since these headers are not in general visible to users they do not need to support encoding in multiple character sets.

The XML DTD for CIM [2,11] introduces a number of XML element names. Similarly these are not visible to an end user and do not need to support multiple character set encodings.

The CIM model [1] defines the subset of the Unicode character set that can be used to name CIM elements (Classes, Instances, Methods, Properties, Qualifiers and Method Parameters). In general these appear as the value of XML attributes or as element content, and in general would not be displayed to end users.

Negotiation and notification of language settings is effected in this mapping using the standard Accept-Language and Content-Language headers defined in [7].

Appendix B

CIM Core Model V2.15

(Can be found on DMTF website : www.dmtf.org)