A Framework for Modeling and Reasoning about Network Management Resources and Services to Support Information Reuse

John Strassner, James Won-Ki Hong, and Kyo Kang
Pohang University of Science and Technology (POSTECH), Pohang, Korea
{johns, jwkhong, kck}@postech.ac.kr

Abstract

Most service and network management applications are developed to use vendor- and device-specific management data. These data are produced from different languages, and hence can have different representations of the same concept. We define a novel knowledge representation and mapping mechanism that uses ontological concepts and relations to generate a formal description for the Directory Enabled Networks new generation information model. This endows the model with the necessary semantic richness and formalism to represent different types of information for use in network management operations. Data is extracted from models, and is used to represent facts. These facts are semantically related to concepts and relations from one or more ontologies using a lexicon. A new data structure is then built by combining knowledge extracted from model elements with knowledge extracted from ontological elements. Semantic relatedness measures are then used to associate modeled data with ontological data. Early results are also presented.

Keywords: information integration, information model, information reuse, ontology, semantic relatedness

1. Introduction

Most network management applications use data obtained from vendor-specific Command Line Interface (CLIs) [1] and/or data conformant to the Structure of Management Information (SMI) [2][3]. While SMI is a standard, the vast majority of network management data are vendor-, technology-, and sometimes platform-specific. These data are low-level in nature, and for example are used to describe the health of a device interface. There are no standards for representing simple higher-level concepts, like a Virtual Private Network, let alone business concepts (such as a Service Level Agreement), because there are no standard CLI or SMI building blocks at this level of abstraction. This means that there is no standard way to build network management applications, and hence there is no easy way to ensure that the services and resources offered by the network are consistent with the business goals of the organization(s) owning and using the network.

The root cause of the above problems is the *lack of an interoperable knowledge representation and associated semantics for network management.*

The use of an information model helps rationalize the diverse set of vendor-specific data models used. Figure 1 shows such an approach [4], in which a single information model (which is independent of data, platform, language, and protocol) is created that defines a standard set of managed objects, attributes, relationships, and other elements. We use the DEN-ng information model [4][5] for exactly this purpose. A set of formal transformations is then applied to translate knowledge from the information model to a standards-based data model (e.g., a relational database using SQL92). This meets the demand of current Operational Support Systems, which use many different repositories that are each optimized for different uses. Optionally, a second set of formal transformations can be applied to translate the result into a set of vendor-specific data models.

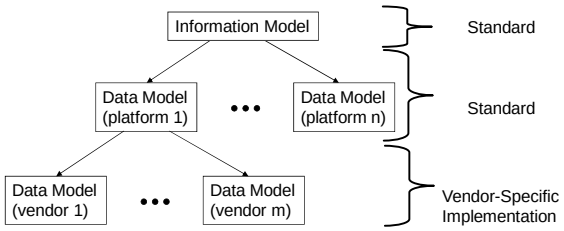

Figure 1. A Unifying Information Model

The usage of an information model helps synchronize syntactical differences. However, it is not able to represent the different semantics associated with each vendor-specific device or technology variation. This is because an information model does not use a formal language, and hence has no innate ability to learn and reason about data.

Context-aware services exacerbate this problem, since the same data can mean different things and require different remediation actions for two different contexts. For example, a configuration management application and a billing application can each define the concept of a "user" in different ways, having different attributes and datatypes; this complicates the exchange of management data and, more importantly, understanding the significance of those data. More importantly, current network management and business support systems have

difficulty relating a term that spans different domains. For example, the term Service Level Agreement (SLA) refers to contractual obligations and revenue for business people, but implies different classification and conditioning of traffic for network people.

This paper describes a novel knowledge representation process that augments knowledge extracted from information models with knowledge extracted from ontologies. This enables the system to construct a machine understandable representation of diverse types of knowledge, and thus harmonize information from different sources to construct a more complete representation of the current context. We use the FOCALE autonomic architecture [6] for managing services using context-aware policy rules. FOCALE uses a *dynamically updateable* knowledge base, meaning that the system is able to add, remove, and edit knowledge at runtime.

The organization of the rest of this paper is as follows. Section 2 briefly summarizes our previous work in information integration and reuse in the context of the FOCALE autonomic architecture. Section 3 describes new extensions to this process. Section 4 discusses our implementation, and Section 5 summarizes the paper.

2. Information Integration and Reuse

The FOCALE effort is developing an autonomic management architecture that can simplify the management processes for the network operator by automating and distributing the decision making processes involved in network operation [6]. In particular, FOCALE uses a combination of models and ontologies to realize information integration and reuse. A simplified architectural picture of FOCALE is shown in Figure 2.

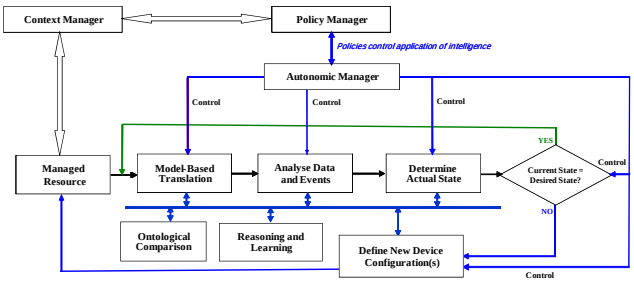

Figure 2. The FOCALE Autonomic Architecture

The FOCALE autonomic network architecture is a model-driven architecture [7]. It can dynamically generate code to (re)configure managed elements; the automation of complex, manually-intensive configuration tasks that are prone to error is a primary motivation for autonomic systems. FOCALE stands for **F**oundation – **O**bserve – **C**ompare – **A**ct – **L**earn – r**E**ason, which describes its novel control loop.

There are three areas of information integration and reuse in the above FOCALE architecture. First, sensor data is vendor-specific, and hence the data from one network vendor device cannot be easily integrated with data from a different network vendor device. Section 2.1 describes how our framework solves this problem. Section 2.2 addresses a related problem: how to translate from a vendor-neutral form of commands to a vendor-specific one. Finally, section 2.3 describes how we reuse information in FOCALE.

2.1 Translating Heterogeneous to Common Data

Sensor data is retrieved from the managed object and fed to a model-based translation process, which translates vendor- and device-specific data into a normalized form in XML. A set of parsers, constructed using the Factory Pattern [8], convert vendor-specific data to a normalized form. This is then matched against data contained in models and ontologies using structural matching and pattern matching, as shown in Figure 3 below.

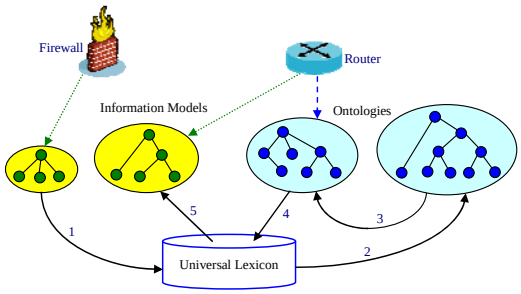

Figure 3. Representing Knowledge in FOCALE

The commands (and management data) of the firewall and router are both represented in an information model. Similar information are represented in associated ontologies. In step 1 of Figure 3, a *lexicon* is used to relate data describing the firewall (in the information model) to a *consensual* set of terms, which are then used to form a mapping between terms in the information model and concepts in the ontologies. Specifically, model data describing the firewall are augmented with meanings from ontologies in step 2 using one or more linguistic relationships, such as synonyms. For example, routers, firewalls, and switches are different types of devices, but share common functions, such as dropping and forwarding traffic. These *generic* terms (dropping and forwarding traffic) can be "attached" to our managed elements *independent of vendor*. Since network management represents a specific domain, we also define a set of custom relationships, such as "is similar to", to perform specialized mappings. An example of this type of mapping is to define the semantic similarity between a set of *m* commands from one vendor with a set of *n* commands from a different vendor.

Most network data is either modeled using information/data models or can be easily transformed into such a model. However, these models, which are built using either SNMP or the Unified Modeling Language (UML) [9], are not able to define semantics, because (1) neither are formal languages, and (2) neither uses formal logic. This also means that UML as well as SNMP models are not able to be used for machine-based learning and reasoning. Ontologies use first order logic or description logic. Hence, we use models to *reuse* existing data, and use ontologies to *reason* about those data.

Step 3 uses semantic relationships (an example of such an algorithm is given in [10]) to relate concepts describing the firewall to concepts describing the router. For example, the verb "accept" can be defined as both a synonym of the verb "forward" as well as an antonym of the verb "drop". Hence, the system now knows that the verb "accept" in the firewall language performs the same function as the verb "forward" in the router language (and also that it performs the opposite function of the verb "drop" in the router language). Step 4 then ensures that these relationships are entered in the lexicon, and step 5 uses these relationships to search for additional model elements (e.g., classes and associations) in the information model for the router. The process iterates as necessary. This knowledge can then be used to generate the appropriate CLI commands that are issued to reconfigure each device.

Data from network devices are structured in ways that make sense to the manufacturer; however, those structures are usually not efficient for information sharing and reuse, and are different for each product. We map raw data from multiple objects into a set of classes in DEN-ng, which uses classification theory [13] to arrange data into a set of reusable object-oriented concepts. For example, the concept of a "card" is applicable to a wide variety of devices, such as servers, firewalls, laptops, and routers, even though those types of devices can be used for very different purposes. An information model builds a set of classes that model the concept of a card as a reusable object, enabling it to be modeled once and then reused for different applications. Different types of cards are modeled as subclasses, as are refinements of a common concept that is specific to a particular vendor or technology.

2.2 Generating Vendor-Specific Commands

In FOCALE, Finite State Machines (FSMs) are constructed from DEN-ng model elements (e.g., classes, associations, attributes, and constraints). Nodes in a FOCALE FSM represent state; each state has an associated set of one or more configuration actions that define the configuration of a managed entity for that particular state. Edges represent state transitions, and imply permission to change the configuration to change the state of a managed entity.

Once the sensor data is translated into a normalized form, it is then analyzed to determine the current state of the managed entity (e.g., a router). The current state of the managed entity is then compared to the desired state from the appropriate FSM. Static behavior is "programmed" into FOCALE by designing a set of FSMs; dynamic behavior is defined by altering the state of one or more managed entities. We use the DEN-ng information model to construct interoperable context-aware policy rule definitions [11][12] to govern the autonomic control loop. This enables context to select the set of policies that are applicable; policies are used to then define the functionality allowed. As context changes, policies change, and system functionality is adjusted accordingly.

In summary, FOCALE defines a normalized network management lingua franca by mapping vendor-specific data and commands to a vendor-neutral form based on a novel combination of information and data models augmented by ontologies. It then uses a model-based translation function to interact with vendor-specific languages and programming models.

2.3 Information Reuse in FOCALE

FOCALE goes beyond the use of models and ontologies to define reusable information. FOCALE supports a *dynamically updateable knowledge base* – one that can reflect new knowledge at runtime as new knowledge is discovered. Our approach supports this requirement by using semantic reasoning to examine sensor data (as well as other types of data) to see if it is new as well as to determine if it is different (and especially, if it leads to different conclusions) than that already stored in the knowledge base. In either case, the semantic reasoning uses first order logic to reason about the validity of the new or changed information with respect to the rest of the knowledge base. If the new or changed information is valid, the system must determine how much of the knowledge base needs to be updated.

The axioms and theories present in the existing knowledge base are used to validate if the new or changed data makes logical sense. This makes use of existing data and relationships in the knowledge base to build assertions and other types of queries to test the implications of the new or changed data. Once the new or changed data are determined to be valid, then additional logic checks the relationships of the changed data to see if those data also need to be changed. Similarly, existing axioms and theories are applied to the new data to hypothesize new relationships.

In general, the new or changed data will either be able to be immediately verified through issuing queries that

verify one or more hypotheses about the new or changed data, or they will need further proof. In the former case, the new or changed data are immediately added to the knowledge base. Otherwise, they are marked for verification. This is beyond the scope of this paper; however, the essential point is that this set of processes enables the knowledge base for our system to evolve with experience.

Both this and the model-based translation function use the notion of semantic relatedness [14] to determine the relevance as well as the validity of the sensor information as well as inferences derived from those data. Semantic relatedness enables entities that are semantically related using synonymy (e.g., "bank" and "lending institution"), antonymy (e.g., "accept" and "reject"), and other lexical relationships such as meronymy (e.g., court is a part of government), as well as defined associations (e.g., router uses protocol). Our original work in this area used linguistic analysis; however, this has a high associated degree of computational complexity. We are thus investigating other means, such as using WordNet [15], which provides a set of APIs for computing common linguistic relations, as well as structural matching algorithms. This enables us to move from offline applications, which require on the order of 2-6 hours of computation, to more near-real-time applications.

FOCALE develops and uses a *library of models and coded behaviors*, much as a library of string processing functions is used by a programming language. This library is made reusable by realizing it in the form of objects, supported by both models and ontologies. Library behaviors are associated with the application of policy actions, which in turn are selected by a particular context as previously described.

FOCALE uses the concept of the Policy Continuum [4][16][17], which enables policies written using terminology and concepts for one domain, such as business analysts, to be translated to policies written using a different set of terminology and concepts for another domain, such as programmers. This enables context-aware policies to be used to orchestrate behavior for business goals, social interaction, and other forms of interaction.

3. Our Knowledge Representation Process

We have extended the above process in two important ways. First, we have formalized the process described in Section 2 using graph theory; the nodes of the graph contain knowledge extracted from information or data models as well as from ontologies, while the edges of the graph are defined as semantic relationships that relate model information to ontological information using various semantic and/or linguistic relationships that each have an associated strength that signifies the semantic relatedness of that relationship. We construct a multigraph from model and ontological data. For example, a switch is more closely related to a router than a laptop is, even though all three can forward traffic.

We then combined this structure with the notion of the Policy Continuum to define a new concept, the Knowledge Continuum [18]. This is shown in Figure 4.

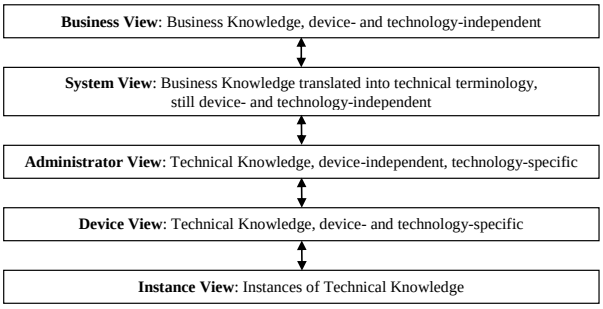

Figure 4. The Knowledge Continuum

Similar to the Policy Continuum, the Knowledge Continuum asserts that in order to ensure the correct understanding of knowledge at one abstraction, and to be able to relate that abstraction to other views, knowledge itself must be represented in a series of views, where each view has meaning in a specific frame of reference, and where each successive view is generated from a transformation being applied to the preceding view. Both continuums are implemented as pure transformation pipelines [19]. This is a forward engineering approach in which any given source model is first restructured in the transformation pipeline according to a formal language and formal transformation theory; this is required in order to prove that the transformation(s) performed preserve the semantics of the models. Transformations are then applied to the resulting formal language, enabling knowledge to be expressed in multiple forms and fused into one common understanding.

Within each level of the Knowledge Continuum, factual and inferred knowledge can exist that is procedural or declarative. Knowledge is assigned to a particular level in the Knowledge Continuum based on whether it is business or technical in nature, and whether it is device- and technology-specific or not. This is used to guide the modelling of knowledge, using information models and ontologies, to ensure that all key concepts from all constituencies in a managed system are represented.

4. Implementation of Knowledge Extensions

There is a profound difference between modeling a fact that is observed or measured and modeling a fact that is inferred. Facts that are observed or measured often do not need additional reasoning performed on them. For

example, if the speed of a device interface is measured to be 100 Megabits per second, that measurement completely defines the fact. In stark contrast, inferences can only exist by having reasoning performed to create them. Thus, our approach defines different representations for each.

Our knowledge processing approach must be capable of representing both procedural and declarative knowledge from a variety of disparate sources, extracting knowledge from those data, transforming the data from each source into knowledge, and harmonizing the result. We define two sets of processes: (1) a transformation of data into information, (2) a subsequent transformation of information into knowledge. This enables knowledge to be defined in a reusable, scalable way.

Data is characterized as observable and possibly measurable raw values that signal something of interest. Data have no meaning - they are simply raw values. Data is transformed into information when meaning can be attached to data. The process of transforming information into knowledge attaches purpose, forms a more complete context, and provides the potential to generate action.

For example, a measured value of 17 is simply a scalar. If that value can be associated with a set of enumerated values to give the value 17 a particular meaning (e.g., "problem"), the system has now succeeded in attaching a meaning to the scalar value. If the system can add further details, such as what the problem refers to and what a possible solution to this problem could be, the semantics are now made explicit, and important additional information and knowledge can now be generated (e.g., a skill set could be inferred as required to solve the problem whose value was denoted as 17). This systematic enrichment of semantics is critical to defining knowledge that can be acted upon. This is reflected in the three top-level hierarchies (i.e., all classes are subclassed from one of these three classes) of the DEN-ng information model in Figure 5.

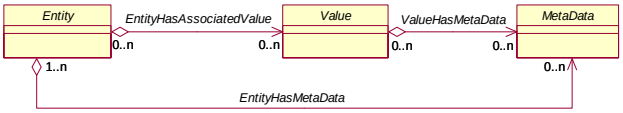

Figure 5. Part of the DEN-ng Top-Level Model

In DEN-ng, a fact represents observations and/or measurements that can be collected, stated, computed, or otherwise proven, and are modeled as classes that are distinct from classes that model inferred knowledge. Facts are represented in DEN-ng as subclasses of Entity, and can be modified by considerations such as accuracy, the age of the measurement, the context of the measurement, who or what measured the value (i.e., a "confidence" factor which could in turn be modified by a "reputation"), and other application-specific considerations. These and other semantics are defined by appropriate MetaData subclasses and associated with facts; this enables different Entities to have different MetaData for different contexts. Note that since Entities and MetaData are both objects, facts and metadata can be reused as appropriate.

Inferred knowledge is modeled using different classes from those used to model factual knowledge. This is because of three reasons. First, facts can be intrinsic parts of an Entity, whereas inferred knowledge is not. Second, the data type and format of a fact are predefined by the type of observation or measurement being performed; in contrast, the data type and format of inferred knowledge can change, because both are dependent on the type of inference algorithm used. Third, since the type and the amount of inferred knowledge can change, DEN-ng uses containers to store inferred knowledge, which enables applications to place knowledge that is computed at runtime into an appropriate container without being bound by a rigid data structure.

Figure 6 shows the six attributes of the class container used to store inferred knowledge. These attributes ensure that data can be stored and understood in an interoperable way. The typeOfContainer attribute defines which type of container is used to house this information, enabling the developer to attach application-specific metadata to different container types. The inferredContent attribute contains the inferred data, and the inferredContentInfo attribute defines how to interpret the inferredContent data. The inferredReferences attribute is an array of strings, one for each reference to an external knowledge source that is required to use this knowledge. The inferredResult attribute defines a set of standard result codes that can be used so that other applications that cannot understand the inferred data can substitute that data with an equivalent result. Finally, the appSource attribute is an array of strings, where each string defines a unique identifier that identifies the application that produced the inferred data. This is useful for tracing the results of the inference operation in case it does not agree with other data.

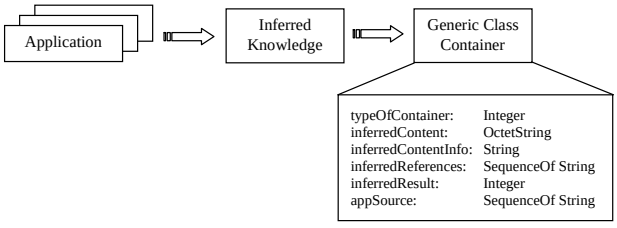

Figure 6. Representing Inferred Knowledge

Currently, our knowledge representation uses graphs and custom data structures to combine knowledge extracted from models and ontologies. This enables existing tools to be used to manage and represent knowledge. Specifically, we use Rational Rose v7.0.0.0 for information modeling, and Protégé 3.4 beta for

ontology design. By transforming existing knowledge from model elements and ontologies into a common format (graphs), it becomes possible to use established linguistic relationships to associate knowledge from model elements with knowledge from ontologies (and vice-versa). This approach lends itself to reusing existing languages, as well as developing new languages.

We chose DEN-ng as the model to integrate into FOCALE because other efforts, such as CIM [20], had a number of problems that make it hard to use with a model-driven approach, especially one that also uses ontologies. For example, CIM does not use patterns or classification theory, so it is very difficult to relate CIM objects to ontological concepts. CIM has its own metamodel which is not UML compliant. CIM has no context model and no metadata model, which are two of the strong points of the DEN-ng design.

Our experiments are currently using different Cisco devices and software releases, which were modeled by extending the DEN-ng model to represent IOS commands as well as hardware, protocol, and other features. We have done limited testing with Juniper and Nortel devices as well to validate our mapping approach.

6. Summary and Future Work

We have described a novel approach to integrating and reusing information of various types for network management applications. This approach is applicable to other domains, and should be somewhat easier to implement, as most other domains do not have the inherent data heterogeneity and programming models that are present in network management. Future work will include reducing our current development environment to a single platform that uses a new symbology that is optimised for knowledge engineering, as well as developing new reasoning approaches that are computational simpler and hence amenable to near-real-time operation.

Acknowledgment

This work is partially sponsored by the WCU (World Class University) program through the Korea Science and Engineering Foundation funded by the Ministry of Education, Science and Technology (Project No. R31-2008-000-10100-0).

10. References

[1] An example of a CLI can be found here (accessed 27 February 2009): http://www.cisco.com/warp/cpropub/45/tutorial.htm

[2] D. Harrington, R. Preshun, B. Wijnen, “An Architecture for Describing Simple Network Management Protocol Management Frameworks”, RFC3411, December, 2002.

[3] M. MacFaden, D. Partain, J. Saperia, W. Tackabury, “Configuring Networks and Devices with Simple Network Management Protocol (SNMP)”, RFC3512, April 2003

[4] J. Strassner, “Autonomic Networking – Theory and Practice”, 20th Network Operations and Management Symposium 2008 Tutorial, Brazil, April 7, 2008

[5] J. Strassner, “DEN-ng Model Overview”, Joint ACF, EMANICS, and AutoI Workshop on Autonomic Management in the Future Internet, May 14, 2008

[6] J. Strassner, N. Agoulmine, E. Lehtihet, “FOCALE – A Novel Autonomic Networking Architecture”, ITSSA Journal, Vol. 3, No. 1, May 2007, pages 64-79, ISSN 1751-1461

[7] www.omg.org/mda (accessed 27 February 2009)

[8] T. Cohen, J. Gil, “Better Construction with Factories”, Journal of Object Technology, Vol. 6, No. 6, pages 103-123, 2007

[9] Object Management Group: OMG Unified Modeling Language Specification, OMG, Version 1.5, March, 2003

[10] A. Wong, P. Ray, N. Parameswaran, J. Strassner, “Ontology Mapping for the Interoperability Problem in Network Management”, IEEE Journal on Selected Areas in Communications, Vol. 23, No. 10, October 2005, pages 2058 – 2068

[11] J. Strassner, J.N. de Souza, D. Raymer, S. Samudrala, S. Davy, K. Barrett, “The design of a novel context-aware policy model to support machine-based learning and reasoning”, Journal of Cluster Computing, Vol 12, Issue 1, pages 17-43, March, 2009

[12] J. Strassner, J.N. de Souza, S. van der Meer, S. Davy, K. Barrett, D. Raymer, S. Samudrala, “The Design of a New Policy Model to Support Ontology-Driven Reasoning for Autonomic Networking”, Journal of Network and Systems Management, Volume 17, Number 1, March 2009

[13] J. Parsons, Y. Wand, “Emancipating Instances from the Tyranny of Classes in Information Modeling”, ACM Transactions on Database Systems, Vol. 25, Issue 2, June 2000, pages 228-268

[14] A. Budanitsky, “Lexical semantic relatedness and its application in natural language processing” Technical Report CSRG390, University of Toronto, 1999

[15] http://wordnet.princeton.edu/

[16] J. Strassner, “Policy Based Network Management”, Morgan Kaufman, ISBN 1-55860-859-1

[17] S. Davy, B. Jennings, J. Strassner, “The Policy Continuum – A Formal Model”, in Proc. of the 2nd International IEEE Workshop on Modelling Autonomic Communications Environments (MACE), Multlicon Lecture Notes – No. 6, Multicon, Berlin, 2007, pages 65-78

[18] J. Strassner, “Enabling Autonomic Network Management Decisions Using a Novel Semantic Representation and Reasoning Approach”, Ph.D. thesis, 2008

[19] E. Posnak, R.G. Lavender, H. Vin. “Adaptive pipeline: an object structural pattern for adaptive applications”, 3rd Pattern Languages of Programming conference, 1996

[20] http://www.dmtf.org/standards/cim/cim_schema_v2210/