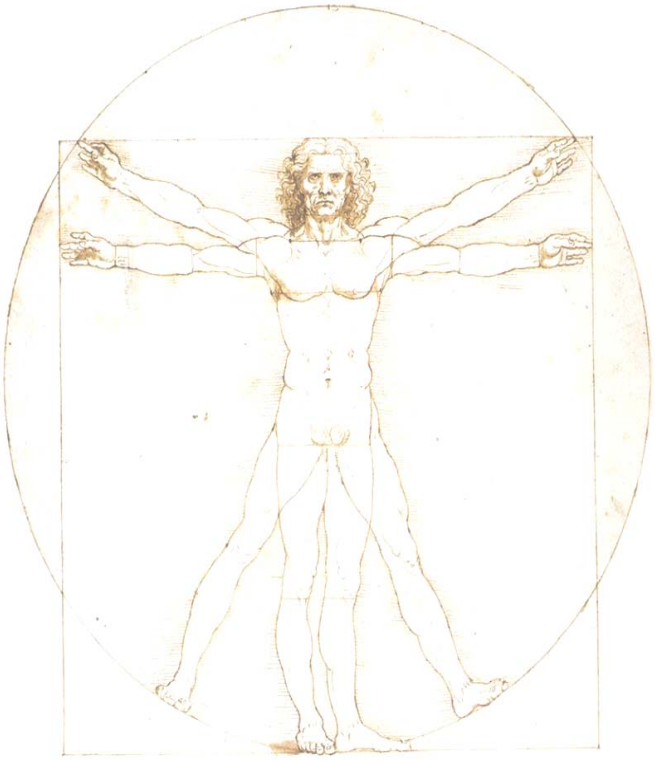
Formal Systems for Interoperability

Welch Allyn Communications Protocol Primer



Science and Technology Office - Technology Primer

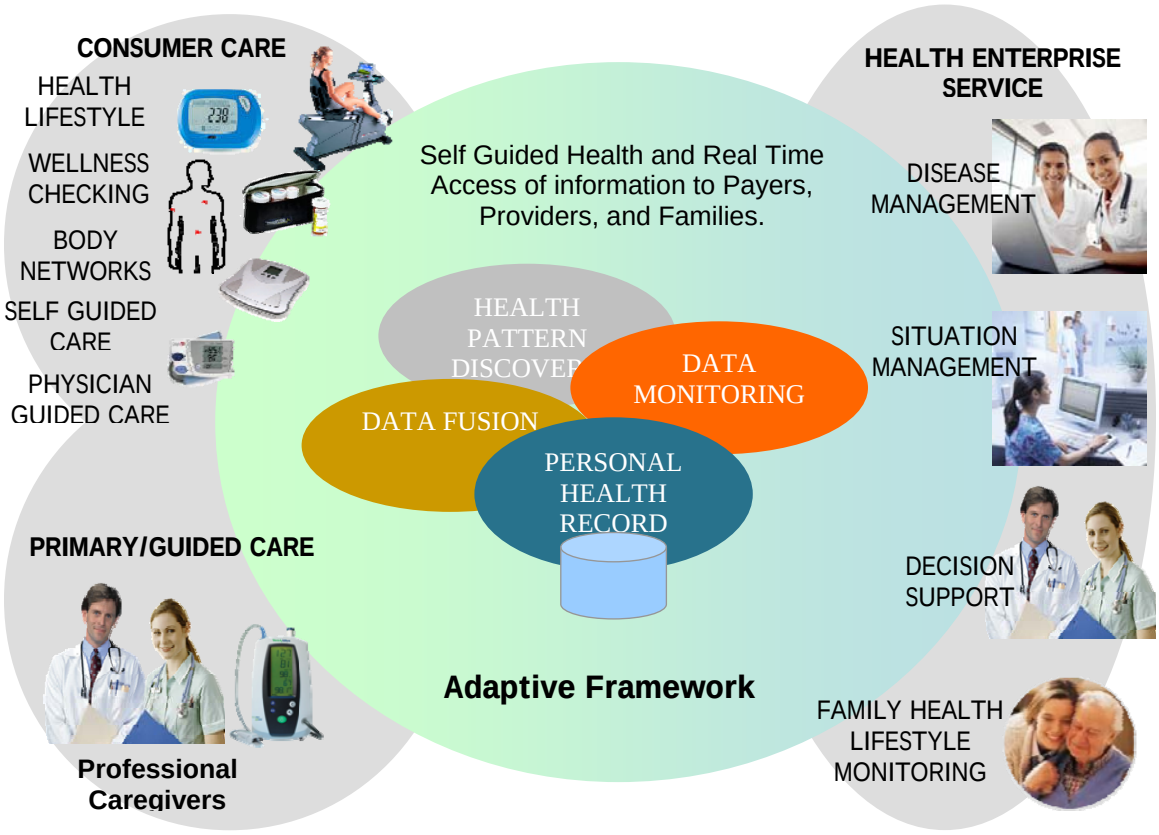
James DelloStritto

The **Vitruvian Man** is a world renowned drawing with accompanying notes created by Leonardo da Vinci around the year 1492. The drawing is often used as an implied symbol of the essential symmetry of the human body, and by extension, the universe as a whole. Symmetry is a precise and well-defined concept of balance or "patterned self-similarity" that can be demonstrated according to the rules of a formal system. A formal system has a formal language, which is composed by primitive symbols. In general a formal system provides an ideal language by means of which to abstract and analyze the deductive structure of information apart from specific meanings and implementation (ref: Wikipedia.org).

Executive Summary

Shifting Healthcare Landscape

The figure below represents an emerging shift in medical practice. The paradigm shift depicted is a trend towards a customer or consumer driven model of self-guided healthcare. This new frontier will require interoperability and seamless infrastructure. Proof of this shift can be witnessed in the use of the internet and other technology. The internet is providing a mechanism for self-guided healthcare allowing patients to play a more active role in their own health management. Technologies like the internet have provided further impetus to the concept of a personal health record (PHR). Other technologies including sensors and wired and wireless infrastructure improvements will further drive health care out of expensive facilities and into more remote settings.



The PHR will likely be the electronic hub for this new paradigm and represents a potential explosion of opportunity in an industry not prepared to meet the demand. The PHR is an electronic record of health information managed by a patient and shared with those they wish to share it with. Health and lifestyle management will become an important tool to improve quality of life, reduce costly hospital stays, and prevent potential harm and/or death. The system in the figure illustrates a future healthcare environment as evidenced by current trends enabled by improving technology. A consumer health market may include healthy lifestyles, wellness checking, body networks, self-guided and provider prescribed care. The consumer market may further include lifestyle management, disease management, and health education/behavior

modification programs. Consumer market applications will be home-based or remote and will require infrastructure and adaptive interoperability due to the myriad of applications and providers across continuous healthcare.

Families will be able to manage their own health. Consumer market applications will be self-initiated or prescribed by a healthcare professional especially in cases where disease management and monitoring are required. The role of the professional caregiver may evolve in to a healthcare partner or manager of an individual's PHR or an entire family PHR. Primary caregivers will likely provide support for the management of the individuals PHR directly and/or through a mobile health assistant like a PDA; or via a PC on the internet. The PHR will contain information garnished from monitored health data. In addition to monitoring and storage of patient data in the PHR, the combination of information in a central location will provide from information (data) fusion, health pattern searches leading to the discovery of positive health trends and trends leading to adverse outcomes. Centralized data may provide for the concept of the 'soft' device". A 'soft device' is the fusion of data and interpretation via an algorithm capable of detecting adverse health outcomes thus providing an indicator to correct and avoid adverse and potentially costly outcomes and death.

Formal Systems - An Interoperability Enabler

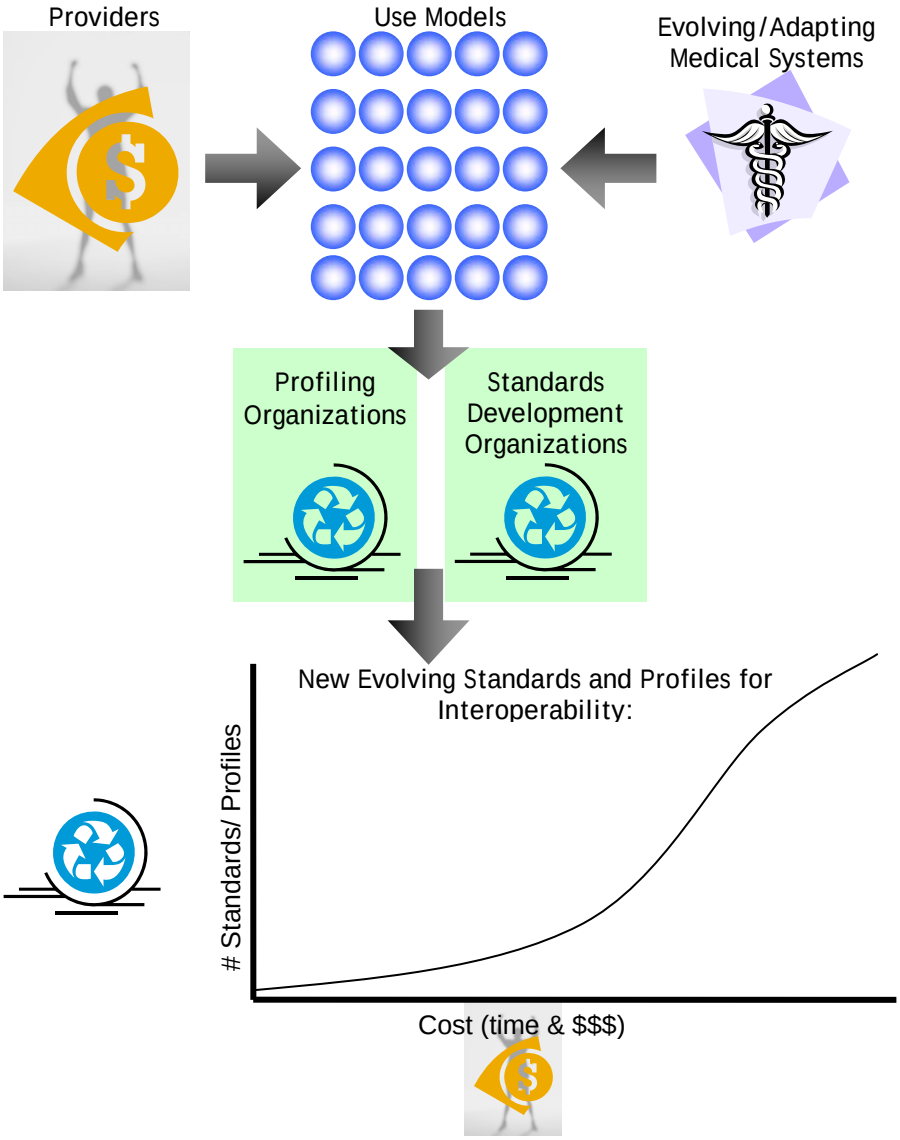
Interoperability will be required to enable the existence and usefulness of the PHR and monitoring solutions. A broad range of information will be required from many inputs in addition to traditional physiologic measure. The subject of this primer is the technology in terms of a formal system. A formal system provides an ideal language by means of which to communicate, abstract, and analyze the deductive structure of information apart from specific meanings and agreed upon standard structures. In this way a formal system provides the language and means in which to communicate, aggregate, abstract, fuse, and inference health states from the complex and varied structures of information found/implemented by the seemingly infinite number of providers in a continuous healthcare paradigm.

A formal system allows for a flexible PHR providing for service-based systems in a health enterprise. In the previous diagram, service-based systems may include disease management, situation management, decision support, and family health and activity monitoring.

Interoperability will be required to allow for the communication and aggregation of the information in a PHR from numerous sources. Interoperability for devices, systems, and consumer products such as cell phones and lifestyle devices including health and fitness equipment can be built independently as long as the rules for the formal system are adhered to.

A formal system provides for a flexible solution to interoperability where each player can innovate and still interoperate with out costly development of agreed upon data structure in standard development organizations (SDOs).

The figure below represents a current approach to interoperability via standards and profile management. There are numerous opportunities for medical information systems improving healthcare (cost, delivery, management, and outcomes). Fully realizing the explosion of new opportunity will require many collaborating industries. The medical system will continually evolve and the technology employed must adapt with change. Turning system opportunities into implementation will require providers converging with medical professionals to define use models in the medical system. Providers today must develop standards and profiles creating a large intricate pool for agreed upon data structures and exchanges and the profiles for medical workflows ensuring manufacturers can interoperate. The diagram below details the cost outcome of the current approach. Interoperability will improve but cost will only be transferred to the endless cycles spent in SDO and Profiling organizations and further reflected in the end cost to the customer.



Standards and Profiles will only multiply with each new opportunity. Cost and other factors like bureaucracy and competitive barriers prevent many providers in a particular use model to participate in an SDO or profile. This only enables the proliferation of proprietary solutions. Standards development counters innovation by requiring companies to agree upon a single structure for data interoperability. Current standards approaches homogenize information into inefficient models that must be continuously revised over time. To evolve existing standards the same players, providers and profiling organization will be required to reconvene. Existing standards need to be continually managed as medicine evolves. Furthermore, medical device manufacturers must adhere to regulations. Regulatory requirements in addition to slow standards process will only exacerbate the delay and cost. Current standards practice in the medical space is unable to empower the inevitable change that needs to take place in healthcare. Standards management may be a barrier to realizing a solution to cost effective and agile interoperability.

Formal Systems in Framework Implementation

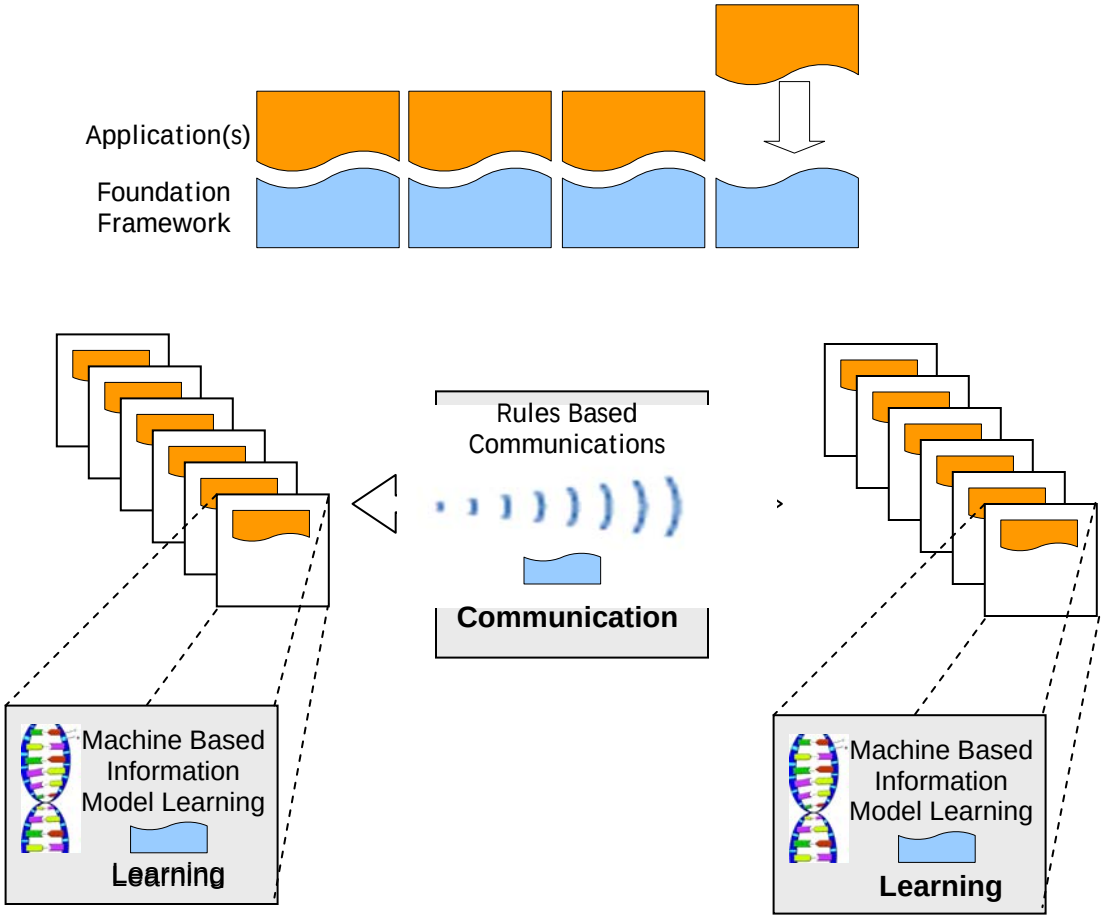
A framework solution provides a flexible base technology to allow for independent development of interoperable information without locking providers into agreed upon data structures. In this context, a framework is defined as a reusable structure for organizing any solution and any information provided by the solution. A framework can be defined in a single standard or minimal set defining levels of optional complexity.

A communication protocol framework defined in terms of a formal system provides a core framework for data representation and message communications in a consistent and extensible way regardless of application information specifics. A dynamic framework nature will provide for information plug and play via an interpretation engine or virtual machine that can learn or perceive new structures of information regardless of the source of the definition. This allows for players in the system to define models of information that are differentiated while still supporting the interoperable environment. It further provides for independent evolution of devices with out cost recompiles and upgrades to existing systems. The costly ripple effect of change is caused by the contract defined in fixed structures between communicating machines. As soon as one machine adds a new parameter the contract is broken all other interacting machines must be rebuilt to handle the changes. This is costly and disruptive. A virtual machine supporting learning will be able to learn/perceive new and changing information models as they are introduced.

This capability decreases the cost of maintenance by providing flexible interoperability as well the freedom to differentiate. This freedom to differential will foster a renewed focus on competitive innovation in both systems and devices. A framework approach removes the cost and competitive side effects in traditional standard development organizations by focusing on the framework and not the application. All providers may build interoperable applications without requiring standards development organization involvement where traditionally a small percentage of companies attempt to agree on a fixed representation for a single parameter set.

The diagram below represents the concept of a framework and the core concepts defined in this primer. Applications are coupled to the foundation framework. There are two framework attributes. The first attribute is communications. The communications framework is a specification for the extensible modeling of information and the rules used to communicate the information in an open environment. The first framework characteristic provides a foundation for the second.

Learning/perception is the second attribute and is defined as the ability for one network device or machine to communicate its information model to another by providing a standard dictionary to an interpretation engine capable of deciphering it. The advent of new introduction of a device or machine to a network would entail an exchange of a dictionary providing a common description for how data is formatted in the device. This dictionary or structure map can be sent by the machine or it may be associated to the machine whereby the dictionary is received from another location and applied upon the advent of the introduction of the new device. The interpretation engine uses the dictionary to extract the information and act on it.



Acting on the received information may depend on the goal of the application. It may display, store, combine and/or fuse the data with other data, run algorithms, or forward the data. To act on the data in this way the data dictionary contains attributes allowing it to perform these functions. This may include display strings, units of measure, references to known medical nomenclature specifications, and any other attributes used to act on the information The dictionary may provide many attributes used to aid in the processing of information.

The embodiment of a formal system in the context of the technology defined here is multifaceted. It provides a language modeling mechanism where by any machine or information set can be modeled. Inherent in the model is the information and the message rules for communicating (a formal system). Simply stated the communications framework is implemented in all machines and each must support the dialog transactions specified.

The framework further manifests in the ability of the information model in the machine to be represented as a communicable data dictionary that can be deciphered by any other machines using an interpreter of the data dictionary. This will enable information perception or true information interoperability without having to agree upon fixed structures. Traditional SDOs only homogenize information and is counter productive in a terminology rich environment.

The framework technology built on the concepts in a formal system enable the proliferation of application technologies as depicted in the figure above without extensive standards management. A formal system supporting learning will provide a level playing field for all enablers in a complex system and the realization of improved lower cost of healthcare without translation of that cost to the management of fixed structures in SDOs. For corporate efforts the technology allows for self supporting systems and a foundation of software to build intelligent products and systems more rapidly.

Preface

The content in this primer gradually increase from simple definitions of a communication system through to the more complex technology. In each real world model the key specifications for a comprehensive communications technology are identified. This is followed by he identification of enabling specifications.

The conclusion following categorizes the technology in the five specifications making up WACP.

The next section details each specification. In these sections a description is provided as well as the theory (if applicable) behind the design. Examples may also be included to entice the reader with the potentials for the usage of the technology and the advantage it provides. The final two sections detail the tools and implementation of Dynamic MOIB.

Real World Models for Communication Systems

The models in this section serve as examples of real world every day communications systems that we take for granted. These models will define and set the stage for the discussions and specifications used in Welch Allyn Communication technology.

Communications systems can be defined in very simple terms. There are three required parts that make a communication system work. First a language must be defined for communicating parties to understand one another. Language combines to provide the first two parts in our communication system; symbols define information and grammar defines the rules for communicating the information. The third required part is a vehicle or conduit to transfer messages and information. These three categories are the basis for any communications system to function effectively.

Model 1 - A Postal System:

Electronic communications can be defined in real world terms by drawing a comparison to a postal system in a three step process.

Step 1: Information

A postal system's goal is to communicate information. Information in the postal system is letters written in a language supported by the sender and the receiver. *Language is the specification that exists in order for the source and destination to carry dialog. Language is defined as a system of common symbols and rules for the exchange of information in a dialog.*

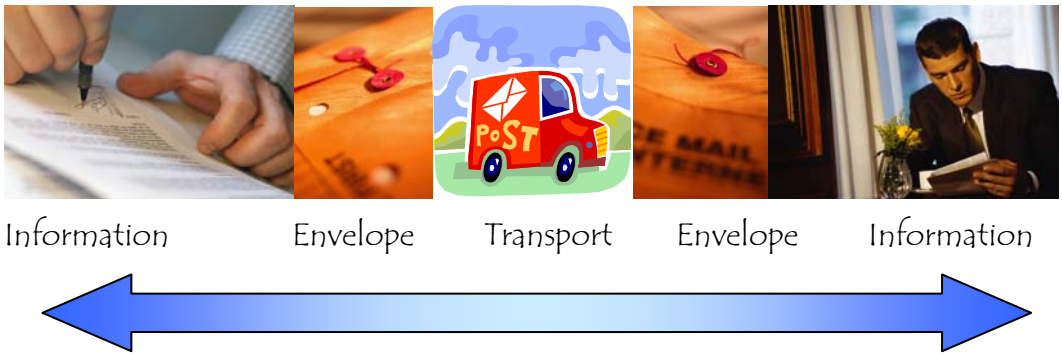
Step 2: Enveloping

In order for the information to be sent it requires an envelope so that it can be transported to the intended recipient. An envelope serves to separate the start of one information segment or letter from another. *Enveloping is the specification for the separation of messages providing dialog control.*

Step 3: Transport

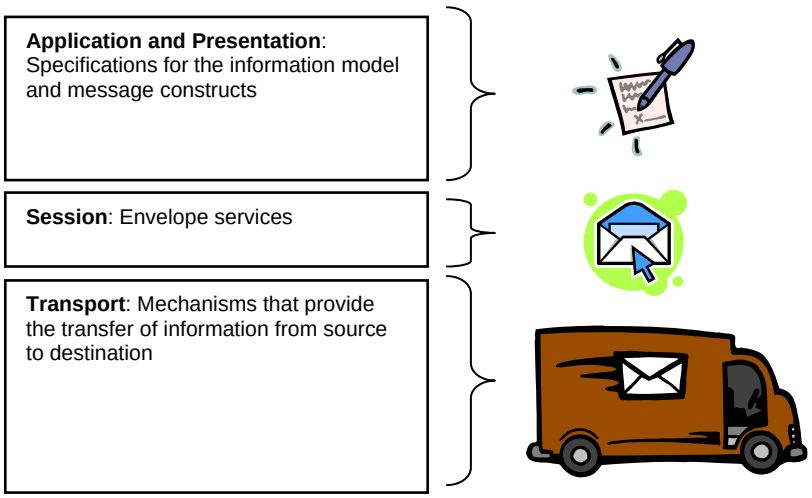
The final step is the transport of the information to the destination. In a postal system this is accomplished by many means; ground, air, sea, etc. A postal system has a specification that defines a method for addressing so that the data moves from its source to its intended destination. *An address written on a physical envelope and vehicles are used to facilitate the proper routing/transport of information.*

Combining each of these steps produces a working communication system.



The postal system is a very simple example of a communication system and is very similar to the methodology employed in any communication system.

The diagram below defines a communication system in levels. Programmers call these levels, Open Systems Interconnect (OSI). OSI is a reference architecture facilitating the creation of flexible network mechanisms allowing information to be easily carried on any transport.



The top two layers define the information model and the means to converse intelligently via the definition of language. The session layer provides the envelope, segmenting messages into distinct consumable letters so the receiver knows the start and end of each. The bottom four layers provide the transport. Like a postal system using multiple vehicle transfer types, computer networks support a variety of physical wired and wireless communication transport techniques. Throughout this document the simple model will be referenced to provide the proper context for the more technical discussion.

Model 2 – Telephone Service:

The second model is a telephone. Technologists refer to telephone service as POTS or Plain Old Telephone Service. POTS is used here as an example of an interoperable communications system.

Step 1: Information.

The telephone service's goal is to facilitate communication. The information in telephone service is governed by language symbols that represent the definition of information in the universe as defined by us. We combine symbols to define our universe in words. Like the postal systems use of written language, telephone service uses spoken language as the specification in order for the source and destination to pass meaningful information. *Language is the specification that exists in order for the source and destination to carry dialog.*

Step 2: Enveloping

Enveloping in human Dialog is the construction of sentences used to facilitate dialog and information exchange. A message envelope may be a simple sentence or group of sentences intending to inform the receiver in the dialog. Sentences are marked with punctuation to indicate the message type providing separation between multiple sentences.

- Requesting information. (Noted by the '?' question mark.)

- Declarative information or action (Noted by the '.' period)

A request sentence takes the following form:

"What is the formula for Hooke's law?

A response with the formula would be expected. However, if the subject "Hooke's law" is not understood, the response should be feedback to the lack of understanding of the subject. Communications systems should be designed to work in the same way.

In this example the predicate (verb ='what is') is a request so an 'unrecognized subject' detailing a lack of understanding of the subject would suffice. At this point, information interoperability has failed but message/grammar based communications has not due to the training humans receive on the grammatical rules in dialog. These rules allow us to understand the differences between a question and a declarative statement as well as allow us to respond intelligently regardless.

The most important point to be taken from this example is that an open system of interoperability must exist in order for a basic framework of structured messages to be recognized and responded to intelligently. Systems designed in this way are more flexible and can provide the ability to learn if these rules for intelligent interaction are maintained.

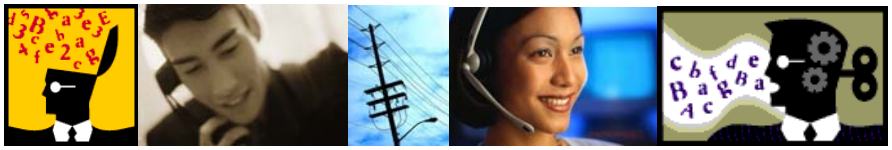
Like the postal system example, Language is the specification that exists in order for the source and destination to interoperate. There are two parts to language… symbol representation and rules that provide us the ability to openly communicate regardless of the subject. Enveloping is accomplished via the rules we follow to create meaningful sentences. In a communication system the interactions are defined as simple sentences that inform, solicit, or command.

Step 3: Transport

In our POTS model, when picking up the phone, the first thing heard is the dial tone. The dial tone is the indication of the successful connection to the telephone service. Dialing a number representing the address of the intended recipient of the phone call will provide the route (like and address on a letter) to the phone you wish to connect to. If entered correctly, the connection can be completed upon acknowledgement of the ring. Acknowledgment completes the connection

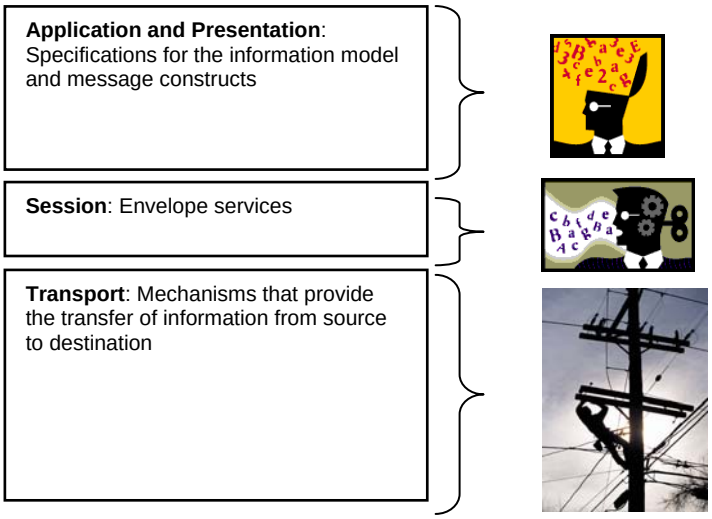
allowing dialog to ensue. *A specification for the transport and addressing to carry dialog is required.*

Plain Old Telephone Service (POTS)



Information Envelope Transport Envelope Information

The diagram below defines the necessary specifications that are required to make interoperability happen in POTS. Like the first example the top layer defines the specification of language used in verbal communications. The second layer provides the enveloping of messages in sentences so that the communicating parties can follow the conversation by listening and responding when grammar dictates. Lastly the bottom layer provides for physical and electrical standards in order for the technology part of the system or transport to work.



The ability to create and understand the meaning of sentences/messages is essential to intelligent dialog. This is one of the fundamental mechanisms by which humans learn.

What exactly does this mean?

A sentence is constructed using two distinct characteristics. The first is the predicate that provides the construct and intent of the sentence. We are trained to decipher the intent of a sentence. We know to listen and respond with a

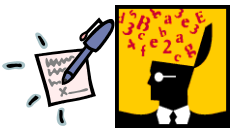
question if the subject is not understood. These grammatical rules allow us to converse intelligently. These same language rules applied in computer language allow intelligent interactions in an openly connected environment. Like humans who can openly communicate, computers must also recognize and adhere to basic rules for interoperability to be realized. This seemly simple ability provides a mechanism for human learning and the potential for machine based learning discussed later in this document.

Required Specifications

In both models above there are a distinct number of specifications required in order for information to be communicated. These are the same specifications WACP employs to accomplish the task of communicating information intelligently in a computer network. WACP combines concepts from the models for a flexible solution.

Step 1 Information

- Specification 1: Symbols (e.g. an alphabet) for Information modeling and representation.
- Specification 2: Rules e.g. (human language grammar) used to create message/sentence constructs.



In the first step of both examples there are two specifications that exist. In both models, language is the specification used for communication. In the first model the language is written and in the second it is spoken. However, both models use language to create messages used to communicate information. Language is defined as the specification of symbols and rules that allow an open system of communication. Open systems are those employing rule based messaging for communicating. This allows constituent members in a network to interoperate openly and intelligently with one another. We will explore this further later in this primer.

Step 2 Enveloping

- Specification 3: Enveloping: e.g. logical separation of dialog parts in physical packages.

In the postal system model the envelope is a physical container for wrapping and moving a letter. In the second model there is no physical envelope. Rather enveloping is implicit in the grammatical rules defining sentence structure. Remember, enveloping is the method used to maintain dialog control. In the postal system model the dialog is controlled and constrained by the disconnected parties that are communicating via letters. The letters contain sentences. In the second model, dialog is controlled by the structuring of sentences. Sentence structure is the foundation of communications in human dialog. Verbal punctuation marks in the form of tone changes and other non-verbal clues mark the start and end of dialog segments allowing us (humans) to openly communicate with one another.

A computer language uses concepts for both models. Computers communicate by creating electrical sentences that include message and information wrapped in an electrical envelope.

A computer language specifies the physical envelope using a string of unique symbols representing the physical start and end of a dialog segment. This string of unique symbols creates the “electronic envelope” allowing the receiver to logically separate message segments and process them in order of reception. Single messages are found in the envelope so they are recognized by all constituents of an interoperable network just as we recognize dialog in the telephone example.

Step 3 Transport

The transport in the postal model is defined as a physical method for the movement of information. The POTS model uses electronic means for the movement of information. The POTS model requires the specification of physical hardware as well as the electrical specification and is similar to computer/device based communication. For instance in POTS a physical connector (RJ11) is defined so that all phones can plug into the jacks found in our homes. POTS also defines the electrical aspects of the communicated voice (voltage levels) and the addressing scheme for locating the receiver with whom dialog is desired.

The Welch Allyn Communications Protocol leverages other transport specifications as they are readily available. WACP does not try to reinvent transport technology. WACP leverages Ethernet, Wireless Ethernet, Serial, USB, Bluetooth, and POTS.

Evaluating the Models

In both real world models three distinct components for communications can be extracted and applied to the OSI model. The first is **Information**. Inherent is the need for message structure supporting dialog and symbols representing the subject in the dialog. The second, **enveloping,** is required to separate messages in to consumable segments. Lastly a **transport** is required to move the information.

These three characteristics form the foundation for the base communications technology in this primer. It also provides the foundation for the intelligence enabling technology discuss later in this Primer. The three specifications derived from our models (defined above) provide the language and the communications transport allowing two parties to exchange information.

Additional Specifications:

There are two additional specifications (implicit in the models above) employed by WACP. These specifications are not required for moving data. Their intent is to enable machines in a network who have information to interoperate more intelligently and flexibly.

Handshaking:

The first of the two additional specifications is similar to the greeting made at the start of a telephone conversation to ensure you have contacted your intended party. When you make a call, the person answering may identify who they are. “Hello, this is John Doe.” In other cases the caller will ask for the intended recipient. “Hello, is Jane Doe at home please?” Computer languages use the same mechanism to identify who they are and whether they speak the same language.

In both models a greeting or salutation would be used to start the dialog. This simple greeting is a confirming communication that ensures you have connected with the proper party. These greeting specifications are often very simple and only serve to ensure that proper dialog can occur. Others may be more complex and include negotiations for security.

- Specification 4: Greeting/Handshake

Self Description:

The next additional specification is a true intelligence enabling protocol supporting machine description.

Imagine for a moment you place a call to a friend with whom you wish to share information with. Let’s say the information is in reference to a new sport that your friend has no prior knowledge of. Before you begin to share information from your experience, you will have to describe the object and characteristics of the game.

In order for that to happen you and your friend will engage in a dialog of questions and information sharing. This process of description is the basis for learning. Comprehension is capacity for the mind to perceive and understand. Medical devices belong to and operate in a highly adaptive information environment. Medical science is a complex system of changing and expanding terminology.

The last specification in the WACP series is one allowing a machine to describe the data and representation of information as defined by it to other constituents in a network. Similar to one person describing a new sport to another in our example above, this specification provides the ability of a device to describe its information model to a consumer.

> “Machine description enables the flexibility of a machine to express its data in a unique way supporting self described interoperability at the time the device connects to a consuming network. This allows machine interoperability with far greater flexibility because each machine is able to describe its information model to other machines on the network. Networks employing this technology can self-correct to meet unmet/unforeseen needs of machines with out requiring recompilation and costly maintenance of contractual structures? “

Some might ask why this is important. Traditionally, simple changes to an information model in a device crippled systems that previously would communicate. This is due to the inability of the consuming machine to adjust its own model to align with another machine’s new or evolving information model.

In complex adaptive environments, information representation changes rapidly as new uses and applications are built. The inability of any machine to indicate these changes to other networked machines is cost prohibitive as well as crippling in the ability to evolve to meet new needs quickly. Consumer applications expect to interact with devices in many different modalities. In order to do so effectively an adapting framework technology is required.

Traditional interoperability in devices is managed by paper standards. These contracts define the form of the data as a fixed structure each device is expected to adhere to in order for it to communicate information. This approach is limited for devices that evolve or are built by different vendors. A vendor may have differentiating information falling outside the fixed set. In cases like this the device manufacturer will likely define or use a proprietary protocol. Paper-based standards limit differentiation and are costly to manage.

Another limiting factor unsolvable by paper based standards is configuration. It is very difficult to create a fixed structure for the configuration of a device. Why? Configuration is often based on hardware. Hardware from device to device is different, therefore the options to configure are likely different as well.

> The final specification is a methodology allowing machines to self-describe their information model and to comprehend new

information or changes to information. This is essential for cost effective non-disruptive sustainability in an evolving and adapting environment.

Later in this primer, models of where this technology might apply will be described.

- Specification 5: Device Description/Comprehension

Categorizing Specifications

The five specifications we have identified are listed below. They are split into two categories. The first three categories are base specifications for enabling the modeling and dialog based communication of information. The third and fourth are Artificial intelligence enabling specifications.

Base Communication Enablers

- Specification 1: Information
    - Symbols (e.g. an alphabet) for Information modeling and representation. Provides the format of electronic data and how to decipher information.
    - Rules e.g. (human language grammar) used to create message/sentence constructs used to solicit and provide information.
- Specification 2: Enveloping: e.g. logical separation of dialog messages in physical packages. Includes addressing.
- Specification 3: Transport: e.g. the conduit in which inform flows with in a network.

Intelligent Communication Framework Enablers

- Specification 4: Greeting/Handshake
- Specification 5: Device Description/Comprehension

These specifications are the 5 core specifications enabling WACP. Built around these specifications are many tools that facilitate the creation of complex models and the automatic generation of software output for use with any Machine. More of these tools and their potential will be covered later in this document.

Welch Allyn Communication Protocol Framework

The Welch Allyn Communications Protocol (WACP) uses the same techniques as the models in the opening of this primer. WACP technology defines the information/messaging model, enveloping, handshaking, self description, and the use of transports allowing devices to exchange Information intelligently. In this section we identify and introduce each specification and define what it is and the theory behind its use.

Specification 1 and 2:

Medical Object Information Base (MOIB) and the Medical Object Management Protocol (MOMP).

MOIB

The medical object information base or 'MOIB' is a classification mechanism used to model information sets representing human physiology or other information.

MOIB defines standard types representing information primitives supported by machines. For example, a primitive might be a simple numeric representing a systolic blood pressure reading or a fixed length string representing the model of a machine producing the reading. MOIB enables information to be represented in aggregate form or as simple types like a numeric. The MOIB technology in very simple terms allows information to be classified in an extensible numeric hierarchy so when communicated other networked machines can identify and extract the information contained in a the package of bytes representing information.

The hierarchy employed in MOIB is the base enabler for the flexibility afforded by MOIB. Medicine is a highly adaptive. Medical applications applied to networked systems are likely to be in a state of constant evolution. As our knowledge deepens there will be shifts and changes in how we describe medicine and medical practice. It is important that a method for defining medical information is extensible and flexible. This is true of any evolving or adapting environment.

MOMP

The medical object management protocol (MOMP) defines the common message rules that allow machines to form the sentences used to communicate the information provided by MOIB. MOMP has a set of standard message constructs forming the foundation of grammatical rules all openly communicating entities in a network must understand in order to carry meaningful dialog.

Just as we can recognize when we are asked a question, a MOMP device can also recognize a question and respond intelligently regardless of the subject. This is also true for other messaging types defined by MOMP. There are five fundamental message types that each machine in a MOMP/MOIB network may support; Request, Response, Command, Status, and Stream.

The Theory

MOIB and MOMP are specifications of a computer language used to define information symbols and message rules respectively. Inherent in these two protocols is the use of a semantic model that defines the grammar or rules (MOMP) used to build common sentence structures as well as the machine information models (MOIB) representing the information machines communicate. We model the computer language in this way so that each device will be capable of intelligent and open communications regardless of the transport used.

Both MOIB and MOMP use the same classification hierarchy in implementation. Semantically the MOMP protocol specifies the message rules governing information transactions and the form it takes on the wire. MOIB specifies the information formats and types supported for transfer and the form the information takes on the wire.

The theory behind MOMP is a computer language modeled after human language. An open system of communication is enabled when message constructs are common to all constituents in a system. As an example, imagine the interaction between two individuals in which one individual asks another for information. The simple ability to ask for information and comprehend a request is the result of our training in the construction of sentences using the rules of grammar. The end result may be a response with the information requested or a response acknowledging a lack of awareness of the information requested.

Machines in networks using the rules defined in MOMP can communicate openly with other constituent members and exchange intelligent answers even if subjects are not recognized. The goal of MOMP is to provide rules to be used by all constituents in a network communicating openly with each other.

MOIB defines the information subject of the intended function of the machine. The main function of a medical device is the definition of information specific to human physiology abstracted from the hardware process retrieving it. Once this physiologic classification is complete it is applied to MOIB. As discussed MOIB is the specification defining physiology or any information model in an electronic form.

The combination of the MOMP and MOIB specifications allow the electronic exchange of data via intelligent dialog.

For example in a blood pressure device, the information is defined using MOIB. Following, MOMP is applied to define the messages that will work with the Blood Pressure information. If a computer asked a blood pressure device for temperature… the blood Pressure device; being trained, will respond signifying it does not support the information sought. However; if the computer asked for the blood pressure, a response should result with the information returned.

This capability is a simple interaction employed by humans trained to openly converse. Asking an individual to give me the equation for Hooke's law might result in an "I don't know" response. Communication comprehension didn't fail. However comprehension of the subject did. The person who asked the question was not trained in subject matter… in this case physics; therefore they are unable to provide the answer.

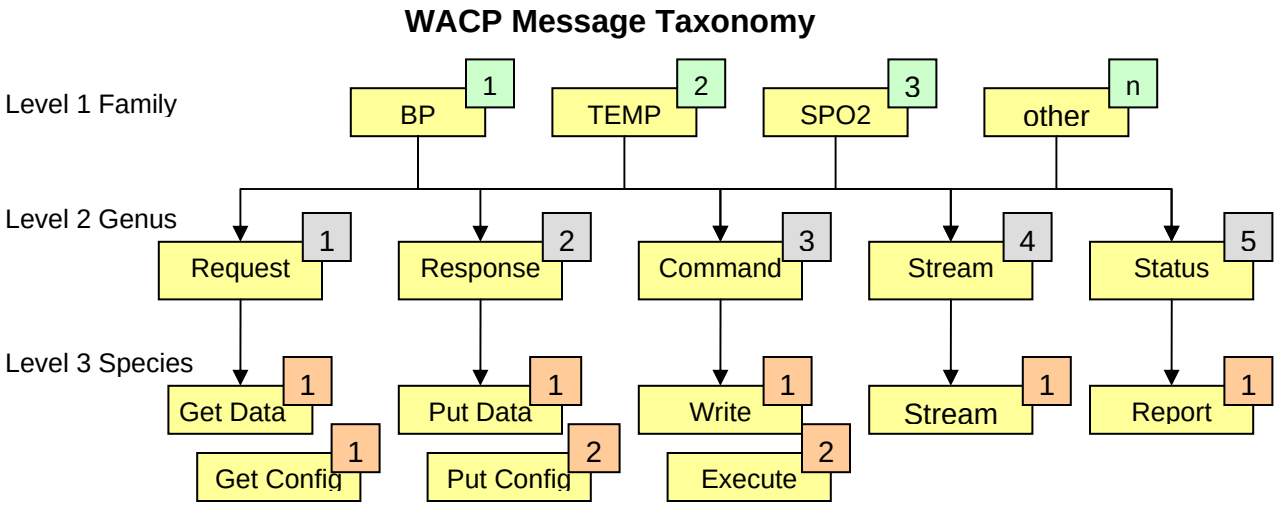
This interaction is the impetus for the learning process. If the dialog above was taking place in a class room a teacher might then proceed with educating the student on the formula. Comprehension is the first stage in the cognitive learning process and is defined as the ability to know, use, or act on information.

The Practice

We use taxonomy as the mechanism to define information and messaging. Taxonomy as defined for biology is; "the science dealing with the description, identification, naming *of organisms*". We apply this to the description, identification, and classification of a semantic model. Semantics is the study of linguistic development by *classifying* and examining *changes in meaning and form*. Notice that both sciences practice classification.

Taxonomy as defined might seem boring but the classification scheme is very extensible. Taxonomy, normally applied to biology, can be applied to the classification of information. It works very well, in that it logically separates information into families that can evolve. This ties nicely to semantic modeling as semantics is the study of ***changes in meaning and form***. Changes in meaning and form best describe the behavior of information models in medical space or in any adaptive complex system where terminology is constantly evolving.

Semantic class based modeling utilizing taxonomy minimizes the complexity of data description management via a design approach that limits, categorizes, and logically groups information management(symbols) and operational functions(messages) into atomic families. The following taxonomic tree is a visual representation used to model messages in WACP.



The first level in this diagram represents the physiological family of information; however, it could be any family of information provided by a machine. The next level defines the semantic identity for a message. Finally the species portrays the specific action the message represents.

A computer language uses numbers to identify messages. Using the tree diagram above we combines numbers in sequence to provide a unique numeric signature for messages in all atomic families. Using the taxonomy example above we can create unique message IDs identifiable by machines engaged in dialog.

1 1 1 = the taxonomic ID for a Blood Pressure data request.

1 1 2 = the taxonomic ID for a Blood Pressure configuration request.

2 1 1 = the taxonomic ID for a Temperature data request.

3 1 1 = the taxonomic ID for a SpO2 data request.

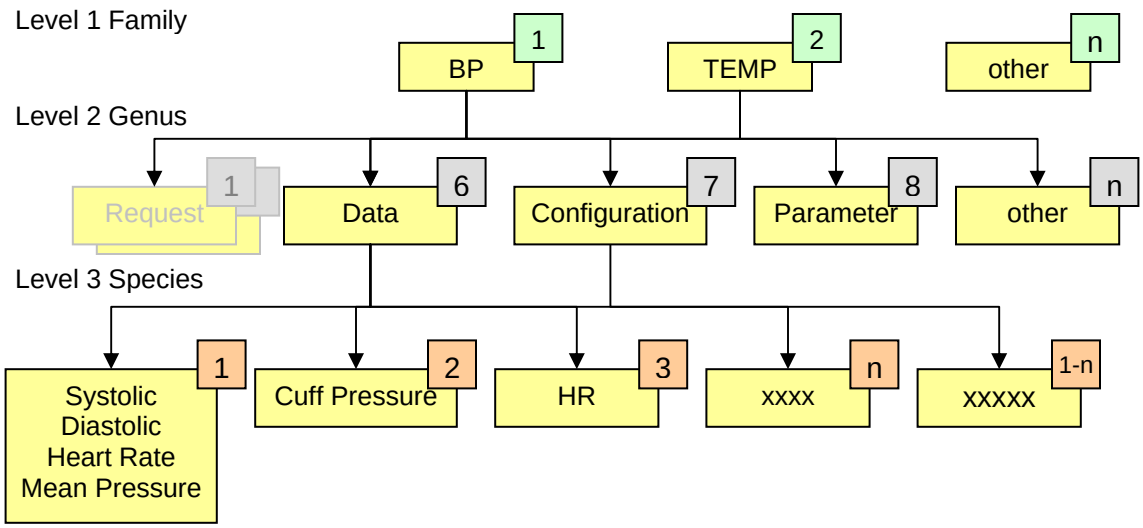
Notice in each family the signature for a request message is the set of combined genus and species {1,1}. Other message signatures manifest as {1,2} for configuration request, {2,1} for data responses, {2,2} for configuration responses, and {3,1} for write commands, etc. The identification provided in the combination of the message genus with the species provides the foundation for the message rules atomic families comprehend in order to carry dialog in an open communication system. The family identifier can represent any family of information supported in a machine. The signature for the message type however, will contain the same signature for any information family defined.

Machines are trained to recognize the message signature regardless of the family. These rules are clearly defined in the MOMP protocol. In our earlier examples we explored the learning process and the need for the ability to understand sentence structure so that we can recognize subject from predicate in dialog. The message signatures defined by MOMP represent the grammatical rules in machine dialog just as sentence structure is defined in human language.

The recognition of the subject from predicate allows a machine to ask for definition when an unrecognized subject is discovered. This ability will be covered later in the discussion of Dynamic MOIB. Dynamic MOIB is the specification of machine based self description.

The message signature defines the predicate or actionable part of the semantic taxonomic model. Next we define the symbolic representation of information in the taxonomic model. The following diagram defines the extension of the blood pressure family to the symbolic representation of information in our taxonomic model.

WACP Data Taxonomy



The first level in this diagram represents the family of information. The genus level is expanded to define the semantic identity for many anticipated and unanticipated information types. Finally the species represents the specific information represented.

Like messages in our model, we use numbers to identify information. In the example taxonomic tree above we extend the model to define the information specific to an atomic family. There are potentially many types of information in an atomic family. Some are sets representing a grouping of information while others are simple types like a numeric. In each level the tree can be expanded as new information is discovered. Combining the numbers from each provides a unique signature or marker for information representation in an atomic family. The

MOMB protocol defines the format of the bytes used to carry the information. The classification number is used as the electronic identifier in the MOIB protocol implementation. All Machines are made of atomic families of information. When combined they provide a taxonomic signature of the device like DNA for living organisms. Using taxonomy and semantics we can model many different types of machine based organisms.

As we demonstrated with message signatures, we can define the information signatures in atomic families. A machine uses these unique signatures to identify data in dialog.

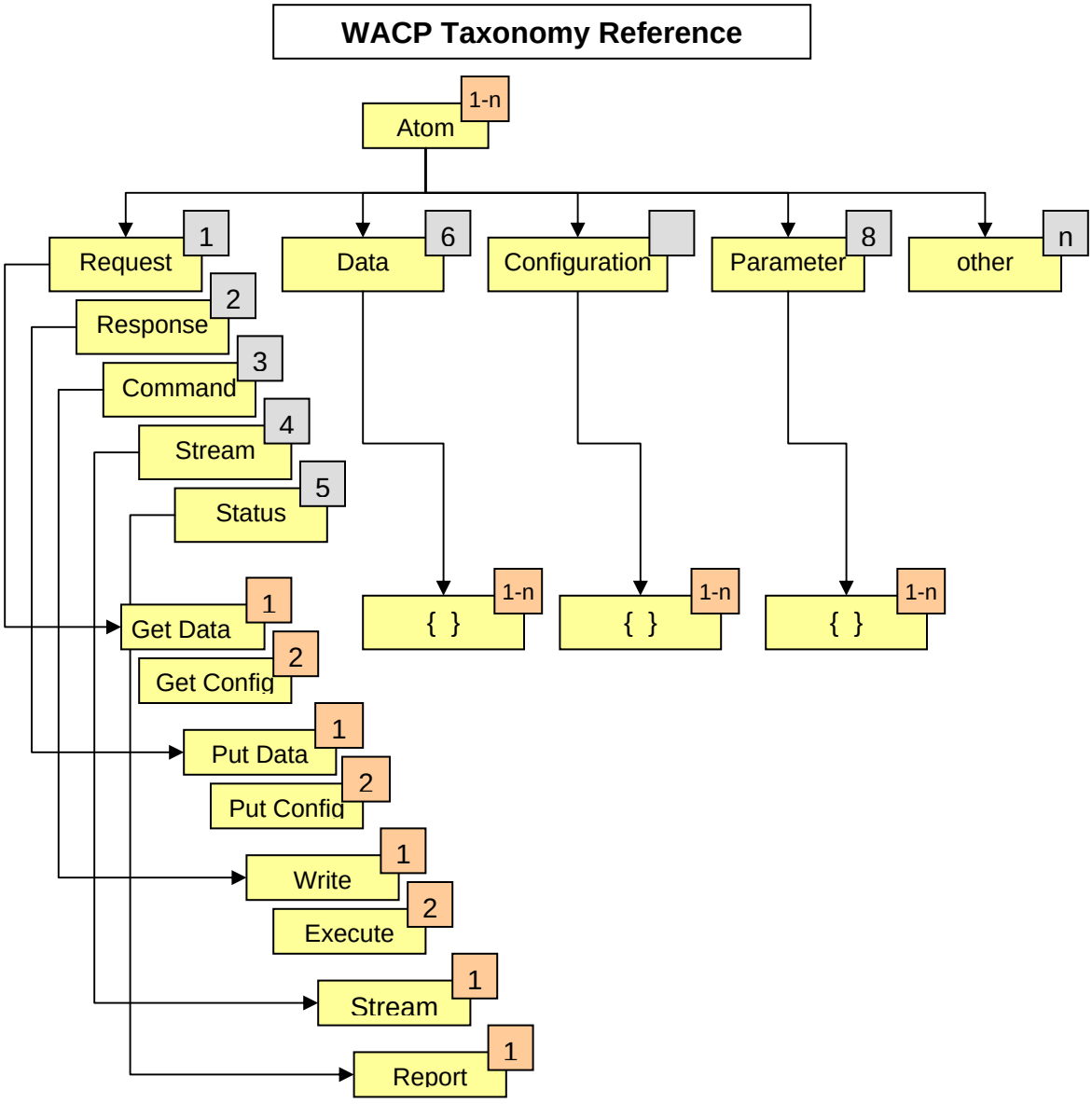
1 6 1 = a complex information type that represents the diagnostic results for a Non invasive blood Pressure (NIBP). Includes the Systolic, Diastolic, Heart Rate, and Mean Arterial Pressure (MAP)

1 6 2 = a simple information type representing a cuff pressure value.

1 6 3 = a simple information type representing a heart rate value as determined by the cuff sensor.

The first two values in the ID represent semantic symbolism for subject. The set {1,6} represents an information part in the atomic family for Blood Pressure. The set {1,7} represents a configuration part in the atomic family for blood pressure. The set {1,8} represents a parameter part in the atomic family for blood pressure.

Combining an atomic family with all of its taxonomic semantic parts will take the form in the diagram below. This diagram is the reference foundation for the taxonomy in the WACP semantic model. All items of information are modeled from this reference.



Historical Challenge: Each level of the taxonomy is capable of expansion to new discoveries and adaptation as information evolves. This capacity for evolution and flexibility is essential. It is important to remember machines will likely have hardware architectures that vary from machine to machine and from manufacturer to manufacturer. This is likely due to customer requirements and marketing requests as well as the engineering capacity and investment in the hardware and software platforms. Due to these differences a single fixed definition is also unlikely to span multiple devices. Hardware may dictate

differentiation in diagnostic physiology definition and more importantly the configuration of the hardware and software systems that capture it.

For example, one blood pressure device may employ more complex hardware/software combination producing a higher resolution pressure reading. The algorithm assisting in the calculation of the diagnostic results may provide configuration options more complex from another manufacturer performing the same function.

Interoperability via the definition of standard data structures or standards is problematic at best. Applications interacting with evolving devices require expensive maintenance updates to understand different manufacturers and device capabilities. Communicating with multiple devices with different information models result in increased complexity in the design of applications. Applications that gather physiology, program devices, or perform other interoperable functions have to be rebuilt when devices information models are changed in device lifecycles. As new devices are introduced, application consumers of machine data must be updated in order to recognize new information models.

The Use

As discussed in the models at the beginning of this primer, WACP has been built based on human language. We also explored sentence structure. Sentence construction is how we communicate. We defined machine message signatures and information signatures in numeric terms so that they can be understood by machines. Machines employing WACP use the numeric signatures to create a sentence consisting of a predicate and subject just as we are trained to do in dialog.

For example we can take an blood pressure example and separate the numeric representations for both the message (rules) and the information (symbols) to create a machine readable sentence. We start by listing the messages IDs:

PREDICATE

These are the available messages for a BP based machine. These messages are synonymous with the predicate or actionable part of a sentence. Next we list some of the information defined in our example blood pressure family; the subject.

1	1	1	= BP Information request
1	1	2	= BP Configuration request
1	2	1	= BP Information response
1	2	2	= BP Configuration response
1	3	1	= BP Write Command
1	3	2	= BP Execute command
1	4	1	= BP Steam (continuous information transfer)
1	5	1	= BP Status

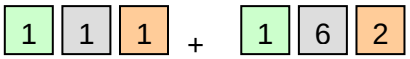
SUBJECT

1	6	1	= diagnostic results for blood Pressure
1	6	2	= a cuff pressure value.
1	6	3	= a heart rate value as determined by the cuff.

Requesting/Response Communications

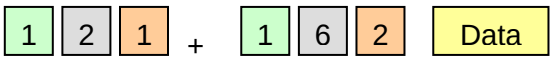
This section defines Request/response dialog and applies to both data and configuration request and response sequences.

Combining the predicate and subject provides a machine readable sentence. A machine readable request sentence for *"Give me your cuff pressure."* Would be the combination of:



Blood Pressure Information Request + Cuff Pressure…

You would expect after asking for the information a return response would result. Using the same method we can combine a predicate with a subject to create the desired outcome. The outcome or response we expect is "My cuff pressure is N". To build a response we need to combine the following.



Blood pressure Information Response + Cuff Pressure

An important note in the process above is that the request normally only carries the signature ID representing the requested information (in this case the cuff pressure). The response however will carry the signature with the attached data (in this case it will be a value representing the cuff pressure). As mentioned earlier MOIB defines primitive types used to carry information. We will look at how this is accomplished later.

In an open network any interrogating machine can manage any other. By management we are inferring the ability to solicit another machine for information openly.

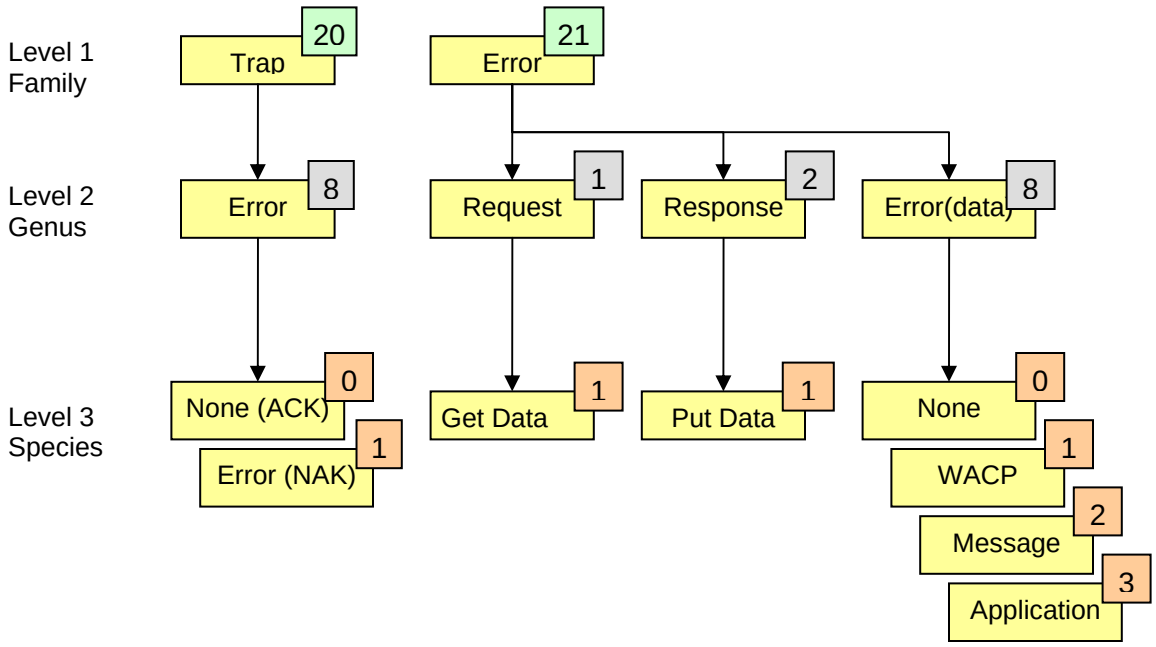
Standard No-Acknowledge

Communications dialog feedback is handled by standard atomic families defined to handle all MOMP message acknowledgements.

The atomic families accomplishing this are the trap and error family. The trap family has the message ability to say "Yes" or "No" in response to any message requiring feedback. It is a common construct used by all WACP machines. A trap is a state indicator for dialog. It affords a simple yes or no feedback to machines requiring feedback from a machine it is engaged in dialog with.

This "yes" and "no" message is also capable of carrying information representing extend information why it can not perform a function. The Error family defines error information that can be carried in the trap message. It includes a machine based number that represents the reason code for the failure and human displayable text defining the failure. In a machine to machine request for data the device solicited for information may return a "no" (trap family message) carrying

information in the form of a text string reading “System Busy” or “Data unavailable” (error family Information). For information requests to a machine not support a family type an “Unsupported family” response results.

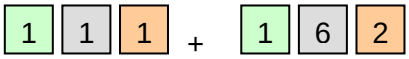


Recall the message signature for a request. A machine receiving the request will recognize the message signature defined earlier. It may or may not support the family ID (?) in the combined signature for the request however.

The machine recognizes the set {1,1} as a question. If it does not recognize the family (indicated by the ?) it must resort to communicating a trap. In an earlier example we defined how human dialog and sentence structure exhibits the ability to recognize the predicate. Information interoperability breaks down in this example however; dialog based interoperability does not.

Here is an illustrative example of how a dialog interaction in WACP is used. The machine to machine interaction in this sequence is still governed by the same rules of language engagement as defined in the MOMP protocol. The request is the same request defined earlier; “Give me your Cuff Pressure”. Each sequence in the dialog is also defined in human readable terms.

REQUEST FOR BP CUFF PRESSURE - (Give me your Cuff Pressure.)

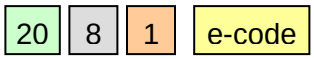


Blood Pressure Information Request + Cuff Pressure…

POSITIVE RESPONSE - (My Cuff Pressure is N.)

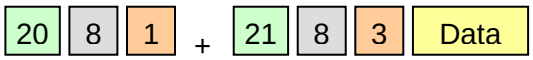
Blood pressure Information Response + Cuff Pressure

NEGATIVE RESPONSE OPTION 1 - (No!)



No-acknowledge (includes an error code)

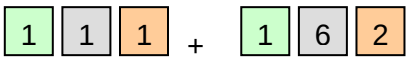
NEGATIVE RESPONSE OPTION 2 (No I am busy.)



No-acknowledge + Reason. (reason contains the code plus human readable data)

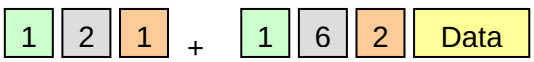
The negative response option 1 includes an error code with it. The error family taxonomy includes a request and a response. This request response will allow a machine receiving a plain “NO” response with an e-code (error code) to solicit the reason from the “NO” response. The following sequence is defined below.

REQUEST FOR BP CUFF PRESSURE - (Give me your Cuff Pressure.)



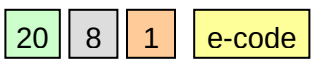
Blood Pressure Information Request + Cuff Pressure…

POSITIVE RESPONSE - (My Cuff Pressure is N.)



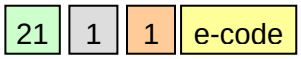
Blood pressure Information Response + Cuff Pressure

NEGATIVE RESPONSE - (No!)



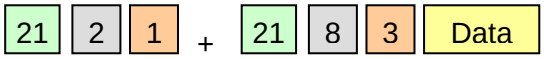
No-acknowledge (This message includes an error code)

REQUEST FOR ERROR INFROMATION - (Give me your Error definition for e-code.)



Error Information Request + Error Number…

RESPONSE - (Error information for e-code is “I am busy”.)



Error Information Response + Human readable error information

Historical Challenge: There is a historical problem addressed by the trap mechanism. The source for accurate error information is always the machine generating the error. In past efforts we have only supplied error numbers from the device and expected the machine receiving the error to map that error to a human readable definition of the error. This has proven burdensome requiring error code maintenance in device hosting machines interacting with multiple devices and versions of those devices. Extensive updates and resolution of error number overlap (e.g. one device uses an error code to define a state different from another device) in systems that rely solely on error numbers as a means to map to human readable definitions.

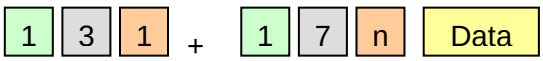
> By leaving the error information in the device and supporting a standard error mechanism to retrieve it, we remove costly updates and extensive error code management from host machines that interact with many devices.

Command, Stream, and Status Communications

Command stream and status messageS all behave similarly. Each carries information. Commands send information with the intent to change the state. Status messages carry information being reported at regular or irregular intervals requiring acknowledgement. A stream, similar to a status message, carries continuously reported information not requiring acknowledgement.

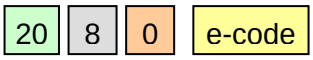
Command Write Message

Just as we did with request and response you can build dialog using the taxonomic model examples. To write configuration information to a device you would combine the following.



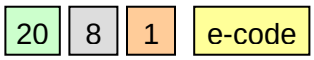
Blood Pressure Command Write + some configuration information…

POSITIVE RESPONSE - (Yes)



Trap (acknowledge or Yes) Message includes an error code likely to be zero.

NEGATIVE RESPONSE - (No!)



Trap (no-acknowledge or no). Message includes an error code.

The trap taxonomy is used for all acknowledgement communication. The e-code returned can be retrieved from the device using the same mechanism as defined in the request response example provided earlier.

Command Execute Message

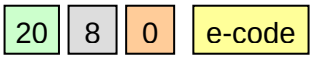
Command execute provides an RPC or Remote Procedure Call function. Execute messages allow for a managing machine to invoke procedures on a device. Command execution functions carry a string based message that represents the function a managing machine wishes to execute on a target machine.

A simple atomic family is provided for the definition of the parameter carried in the execute command. It is a simple information type carrying a string.

In the example below we intend to Start a Blood Pressure. The sentence combination in the execute signature below would be the simple sentence; “Execute the start BP Cycle”. The parameter in the signature will carry the command recognized by the machine. The machine is required to respond with the outcome of the execution command.

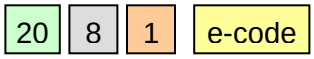
Blood Pressure Command Execute + parameter (“Start_ BP”)

POSITIVE RESPONSE - (Yes)



Trap (acknowledge or Yes) Message includes an exception or zero for no error.

NEGATIVE RESPONSE - (No!)



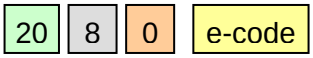
Trap (no-acknowledge or no). Message includes an error code.

Status Message

A machine can periodically report a cuff pressure (or any other information) during the NIBP process. This report can be invoked by a machine based process or from human interaction like the push of a send button on the machine.

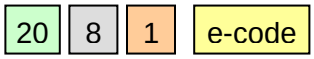
Blood Pressure Information Report + Cuff Pressure…

POSITIVE RESPONSE - (Yes)



Trap (acknowledge or Yes) Message includes an error code likely to be zero.

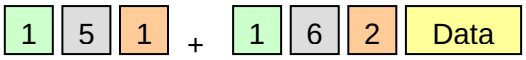
NEGATIVE RESPONSE - (No!)



Trap (no-acknowledge or no). Message includes an error code.

Stream Message

Streamed information is often time critical and continuous and does not require acknowledgment. In the following simple example a systolic reading from an Invasive blood pressure monitor streams to a target monitoring machine at some determined frequency.



Blood Pressure Information Report + Cuff Pressure…

Conclusion

In this section we defined the science behind how we use taxonomy applied to semantics to create a flexible rules based computer language. Using taxonomy we applied numbers in a hierarchical fashion providing a flexible and extensible mechanism for defining atoms of information in a machine. Each atom is extensible at all levels of the taxonomy and can evolve independently over time. This logically separates atoms of information into manageable sets that can be combined to create new atomic configurations of machine based communications.

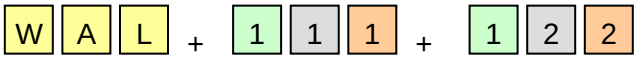
We defined how we use taxonomy for the creation of messages providing a consistent means to govern and enforce the rules of language engagement. This provides the ability for machines to respond intelligently to the actionable or predicate of all messages defined in the computer language's grammar.

Specification 3: Interchange

The Interchange specification provides the enveloping mechanism for messages and data passed between constituent members in a system. It can also track whom it is carrying a dialog with. Similar to the addressing scheme used in the postal system, it is a mechanism for tracking where a message comes from so that the response will be directed to the right place. It works like a physical post office dispatching mail to the proper location.

The enveloping itself is accomplished in a way similar to the mechanisms for creating the electronic sentences modeled earlier before. In our first model we wrote letters. In the letters we punctuate our sentences so that the letter can be read properly. In the second model dialog is spoken and verbal punctuation (e.g.pauses) are used to envelope and separate distinct parts of the conversation.

The interchange specification defines unique strings called preambles. The preamble is a set of symbols that provides an electronic means of separating messages for processing. Building on our previous examples, a preamble in WACP will take the following form.



Preamble + Information Request + Cuff pressure…

This is necessary for instances when many messages are being received. The preamble provides a mechanism by which we easily search for and test the integrity of messages as they are received from transport pipes. It also provides us the ability to separate and queue up messages for processing by applications using WACP.

The preamble also allows us to identify domains of information as well. The WAL preamble is one preamble found in WACP. There is another defined “WAC”. WAL is a light weight preamble marked by the “L” in WAL. It is a simple envelope only intended to carry simple clearly separated messages. Other preambles like WAC contain session based information allowing large messages to be broken up into pieces. These preambles contain additional information including packet ordering. This is like receiving a series of packages to be reassembled at the application layer. Large files can be transferred by WACP. The WAC preamble provides a target machine the clues it needs to properly reassemble the package.

Additionally the preamble can be used to define application domains. This will allow a medical application to recognize medical communications as well as other defined domains. This is beneficial for fire walling messages as well as mixing data from multiple applications.

Specification 4: Rendezvous

Rendezvous is a handshaking protocol that provides a process that first identifies the device as a WACP device (via a simple string) followed by an optional negotiation of encryption keys.

Specification 5: Dynamic Medical Object Information Base

Dynamic MOIB (DMOIB) builds from the other specifications and provides the basis for an intelligent framework. MOMP and MOIB are modeled on language and constitute the base enabler for DMOIB. We established these specifications as required for open communications. Building from open communications, we can determine the full potential of WACP combined specifications with the use of DMOIB.

Learning

Examine the discussion between a teacher and student. The teacher and the student have the ability to communicate using the rules defined in grammar. Each sentence is enveloped by these rules so that dialog can ensue. Statements made during the learning process by a teacher may contain subjects not recognized by the novice due to the lack of comprehension by the student.

The advent of the lack of comprehension by the novice marks the breakdown of information interoperability but not dialog. As intelligent adaptive systems, humans use the rules of grammar to construct sentences which solicit the information required to comprehend the subject. This is the basis for the first step of the learning process, comprehension or perception. The student at this point may ask questions in order to comprehend subjects provided by the knowledgeable participant in the dialog, in this case the teacher.

Machines in traditional networks have been unable to support the basis for learning, even simple perception. If the subject in an exchange of information is unrecognized it would remain unknown and interoperability breaks down. The knowledge deprived machine would have to be rebuilt or reprogrammed with that knowledge. Traditionally, machines have only been able to comprehend that which the programmer instilled in them. These machine entities are simple robots incapable of expanding rules and information to adapt to a fluctuating environment.

In the discussion leading to this section we defined some of the challenges faced in an interoperable environment. Changing information models in machines cause a breakdown in interoperability.

Systems and communications in the past have lacked the intelligence to support the first step in the learning process. DMOIB is an intelligence enabling technology allowing machines to respond intelligently to the changing interoperable environments by supporting the ability to comprehend new information models and entire machines.

Comprehension

DMOIB is an implementation of MOIB providing a machine with the ability to describe the model and representation of information in an evolving network. The example of the student carrying on dialog with a teacher to comprehend is a basic learning process and the theory behind how DMOIB works.

Comprehension is the first step in cognitive learning. The cognitive learning process includes the ability to infer action or recognition from existing knowledge in the advent of the unexpected. Patterns plus recognition of related concepts form the neural pathways leading the ability for intelligent systems to infer. Intelligence is enabled in a system modeled by structural properties whereby behavior and outcomes can be inferred by combining multiple structural properties into a web or network of interrelated decision making steps. This is true intelligence and is fuzzy as can been seen in real world models like living organisms. Human beings exhibit the ability to make judgment errors when making decisions. This is likely due to a complex processing model employed by the human mind. Still the concept is simple. Input is gathered and acted upon in some way. Past experience and knowledge of outcomes and the physical world all play a part.

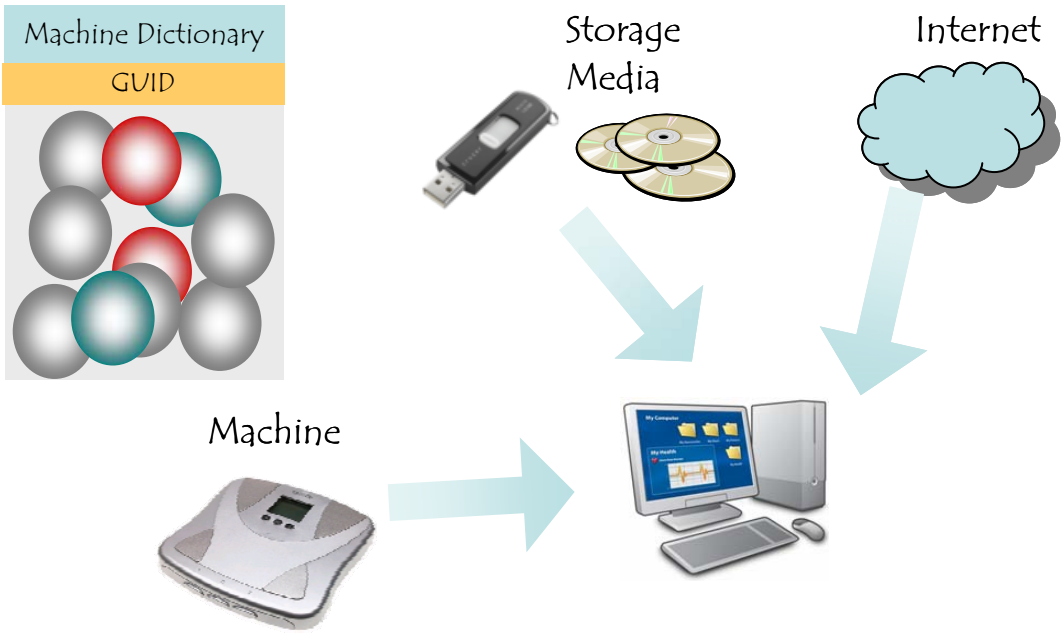
MOIB can be used to create a complex model of an environment. It can then be applied to an intelligent processing machine taking into account the defined attributes of a system in order to determine outcomes and learn from errors in judgment. For example, in order for an intelligent computer to drive an automobile it would require the basic rules of the road and the physical world around it. It also requires the ability to infer the unexpected. If a bowling ball is thrown at the car, the car would need to infer the outcome based on the behavior of the ball and its attributes. It would then need to make a decision to brake based on an assessment of the potential damage that might occur if a collision where to happen. Environmental attributes and behaviors are essential in determining whether it is really a bowling ball, or soccer ball, or balloon for example. MOIB has the ability to define complex models that will allow leaning to occur. This will be further explored later in this document.

Before we explore full cognition and how this technology could be applied we need to explore the capability supported it terms of interoperability and how it is planned to be leveraged. DMOIB supports the first stage of cognition. In logic, the comprehension of an object is the totality of intensions, that is, properties or qualities, that it possesses (*ref wikipedia*). Comprehension provides the means to

enter into the first stage of the cognitive process. Comprehension is the ability to know, use, or act on information.

Comprehension can be defined in a two step process; perception and action. Perception is the ability to take in information via the senses, and process it in some way. Vision and hearing are two dominant senses that allow us to perceive the environment (*ref wikipedia*). Action is taken to refer to the output of a system. In humans, this is accomplished through motor responses. Spatial planning and movement, speech production, and complex motor movements are all aspects of action (*ref wikipedia*).

Applying these principles to a framework for use in machines is the end goal accomplished by DMOIB. The first step in DMOIB is perception. In the section on MOMP and MOIB we defined the modeling practice for defining information in a machine. DMOIB takes the taxonomic model and transforms it into an electronic dictionary. This dictionary is transferred from machine to machine in a simple learning process (perception). Once the data is perceived it can be acted upon. In DMOIB this action takes two forms. The first action is display or storage. The dictionary carries the human readable representation of information including the definition, units of measure, scale for precision of numeric readings, as well as an aka (also known as) for machine based processing algorithms. Providing this simple technique allows for powerful self maintaining systems. The technology provides a mechanism for adapting systems to change and interoperate with any device encountered under any changing condition. The perception phase is the discovery of machine dictionaries. This stage is accomplished in two ways as demonstrated in the diagram below.



In the diagram the machine dictionary or Device Data Sheet (DDS) is an electronic representation of the taxonomic model of a machine. Each sphere in the dictionary is the atomic representation of a taxonomic family. This dictionary is created from the taxonomic model of the device via a tool set discussed later in this primer. The diagram shows three methods for a computer platform or other consuming machine to be trained for a new machine model. The machine may contain the dictionary and transfer it to another machine hosting it. If the machine is unable to store and transfer the dictionary, a CD or other storage media type is used to house the information model. The definition could also come from an internet website maintained by the manufacturer of the machine. The machine dictionary or DDS contains what is called a Globally Unique Identifier (GUID) used to associate a dictionary to a machine when it is provided from a source other than the device.

The Data Dictionary

In this section we will explore the format of the dictionary and how an intelligent machine uses it to act on data. The dictionary provided is very flexible and supports multiple languages via the use of a linking mechanism. Before we define how this works we need to describe the way primitive information is carried in communications. Primitive data are variables transferred in communications and are defined by an agreed upon encoded form. As discussed, MOIB provides a set of primitive electronic types used to encode and carry information in electronic form. The table below defines some of these encoding types and how they are identified in the dictionary. For more information, the MOIB specification has a complete listing.

Identifier	TYPE-SPECIFIED
0x01H	Uint8
0x02H	Uint16
0x03H	Uint32
0x04H	Uint64
0x05H	ansichar (String)
…	

For clarity we need to examine the table above and define how it used. The identifier on the left is a value or number in the dictionary used to identify a piece of information. For example if we define a Systolic value in the dictionary it would likely be carried as a 16 bit integer (numeric value). Likewise the diastolic, heart rate, and MAP values would also be carried as 16 bit values. Using the taxonomy

defined earlier we recall the ID for an aggregated blood pressure reading is the combination:

1	6	1

= diagnostic results for a Non invasive blood Pressure (NIBP)

This reference exists in the dictionary as an atomic reference and is followed by the encoding IDs of the members it carries. In the dictionary the set {1,6,1} is followed by 0x02H, 0x02H, 0x02H, 0x02h. This is a map of the encoded data needed for a machine receiving the atom represented by the set {1,6,1}. The values following define the data carried as “uint16” encoded. These are 16bit numeric values. When an ID {1,6,1} is received the values are extracted. In plain terms the dictionary table describes how to extract information received.

DICTIONARY TABLE - NIBP ATOM						
1	6	1	0x02	0x02	0x02	0x02

Once extracted the data can be further processed for display by using other dictionary attributes defined below. The figure below defines default display text, units of measure, scaling, and AKA for the atom and each individual member (Items 1-n). The Items in the dictionary represent the Systolic, Diastolic, Heart Rate, and MAP values. Notice the units are defined individually for each member and not at the atomic level. If this atom contained a set of values with the same unit the atomic definition could carry units and scale for all members. In our case we have differentiating units so each member item carries its own definition. The diagram below contains information for the atom as well as information for each member.

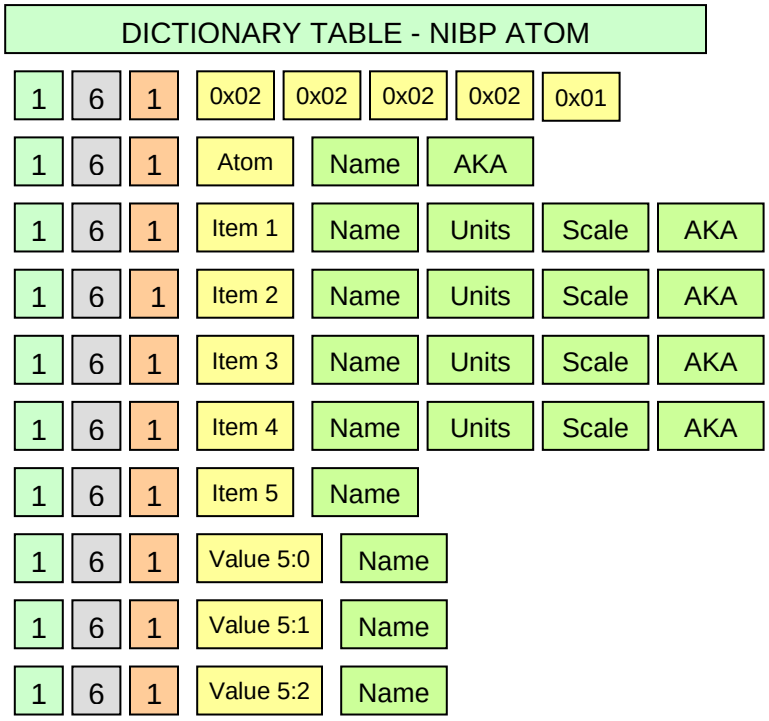
Dictionary Entry for an NIBP result with four numeric values

DICTIONARY TABLE - NIBP ATOM								
1	6	1	0x02	0x02	0x02	0x02		
1	6	1	Atom	Name	AKA			
1	6	1	Item 1	Name	Units	Scale	AKA	
1	6	1	Item 2	Name	Units	Scale	AKA	
1	6	1	Item 3	Name	Units	Scale	AKA	
1	6	1	Item 4	Name	Units	Scale	AKA	

The dictionary also supports enumerated string types. Adding a “signal quality” value to our NIBP entry, we define three states (Green, Yellow, and Red.) The

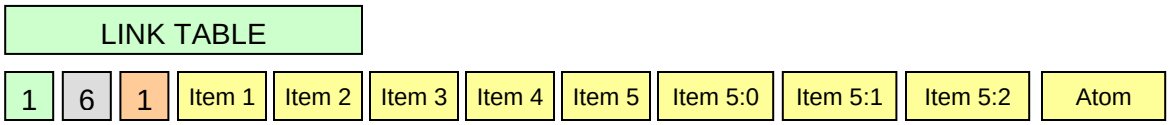
three levels may be an indication of quality determined by factors like signal strength at the sensor. This can be defined in the machine dictionary. We start by adding the member to the original base library identifiers. We use an 8 bit integer for representing the value (item 5 in the diagram below). The encoded value for item 5 is added to the map with a 0x01 value. This is then followed by the addition of the actual line item with name and the enumerated string members represented by Item 5:0, item 5:1, and item 5:2.

Dictionary Entry for an NIBP Atom with Reading Quality

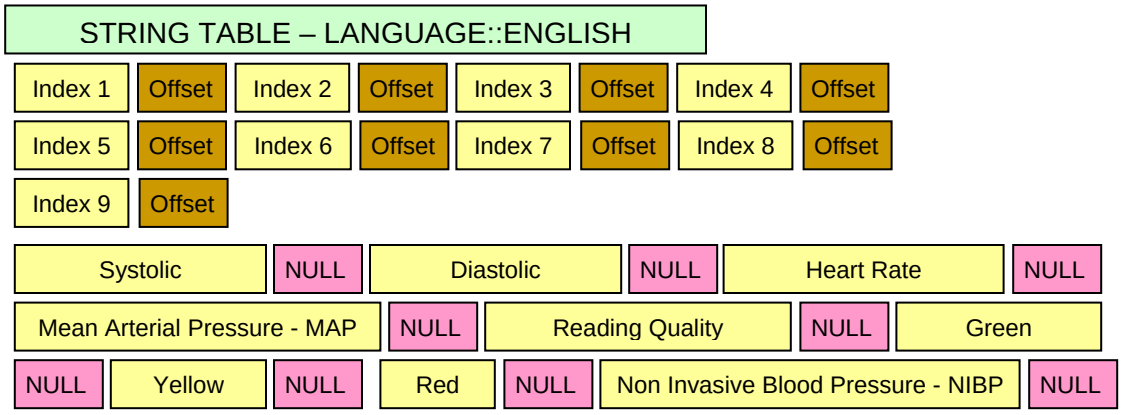


The value carried in the “signal quality” member is a number from 0-2. The dictionary provides the human readable value for the numeric representation. This way a human readable value can be displayed. If the last of the five values carried by the NIBP atom is the value 2, the human readable representation is taken from the dictionary at the value for Item 5:2. In this case it would be Red. This value can be displayed to the user signifying the confidence level of the reading as determined by the hardware capturing the reading (for example).

The definition above is the base library providing the ability to extract machine data and display it using the default language provided in the “Name” attribute. The library is not limited to this. It is capable of being localized for any language via a linked list mechanism. The linking map or table is used to correlate the “names” in the dictionary to any language. Looking at the dictionary above there are human readable “name” entries for each member in the dictionary. We use the Item reference to link to string maps for any language. The first diagram below represents the link table.

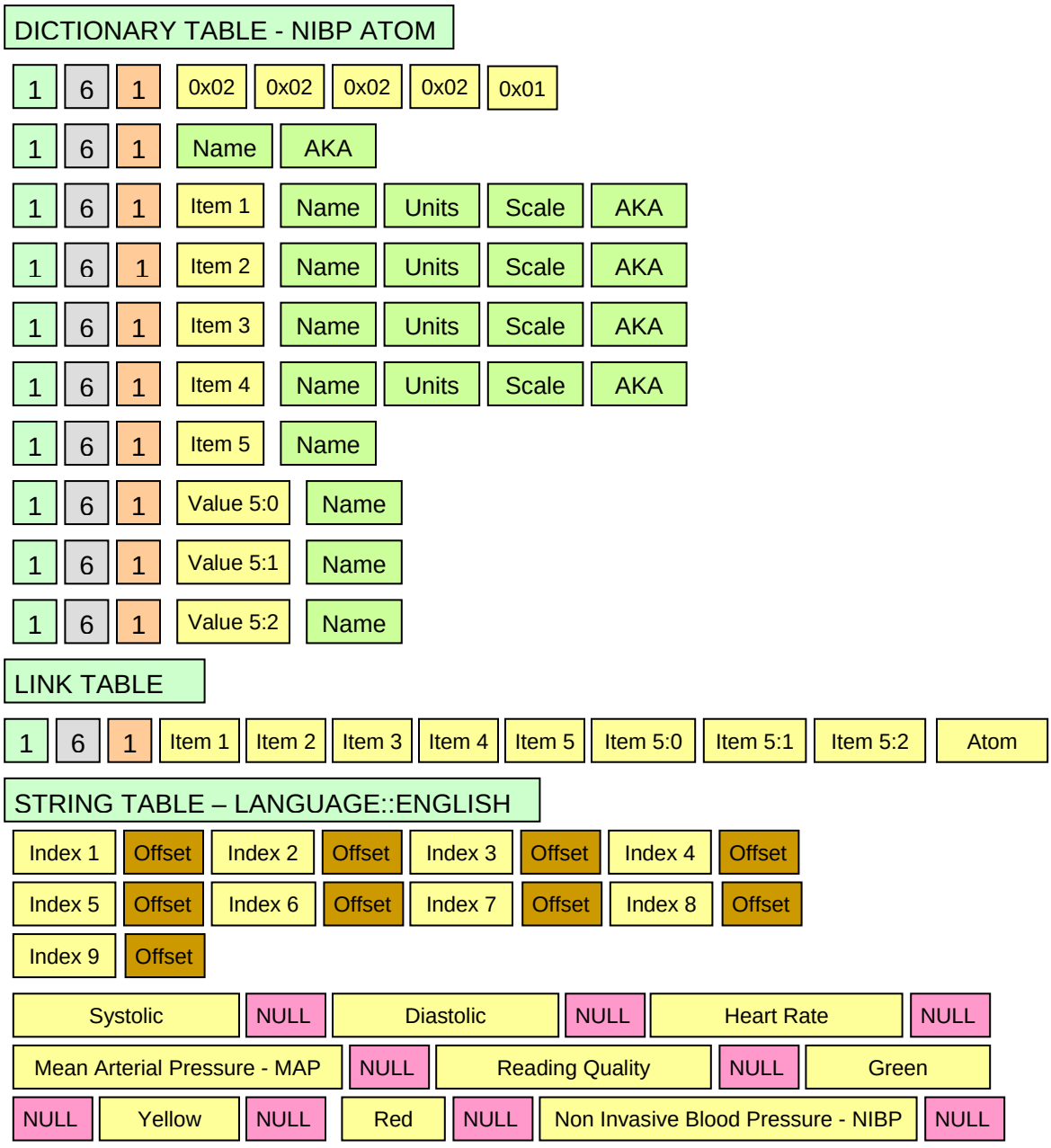


The dictionary relies on the link table above to define the location of localized strings in any string table. The string table is defined in the following diagram. The link table above is ordered and related to indexed items in the string table below. Item 1 in the link table above correlates to index 1 below, Item 2 to index2, etc. The index contains the offset into the attached string table for the localized string.



The string table has been defined separately and then linked for a specific reason. One of the challenges “self described” technology faces is duplication of string display text in the machine. The device user interface (e.g. a screen on a machine) requires its own string representation for all of the languages it supports. DMOIB also requires localized strings in order for the device to self describe itself to applications in any language. The string table defined above is designed to be accessible to both the embedded device and DMOIB. It can be transferred by the device during the self description phase. This eliminates duplication of string table definition in the device. It also provides a consistent string/display definition as defined by the machine. The device defined strings passed during self description are the same employed by the device user interface screens saving device memory, providing consistency, and a higher likelihood the device will employ self description due to the strings being available anyway.

The full dictionary for the NIBP atom is defined below. It consists of the main dictionary, the link table, and one or more language tables.



Looking at the dictionary above we can define the process DMOIB uses to extract and provide data for display or processing. Assume the compute platform has received the dictionary above. When it receives a message containing the ID {1,6,1} it will use the dictionary to prepare/process the data.

Step 1 is extracting the data (assume the dictionary has been received). It uses the *dictionary table* to recognize what to extract. As defined earlier, primitives are defined as electronic representations of variables carried by an atom in MOIB. A value of 0x02h is the ID for a 16 bit numeric. DMOIB uses this information to extract 16bits of information from a buffer of data identified as {1,6,1}. It then

extracts the four other primitives. It now has 5 separated variables representing our NIBP Atom.

Step 2 is the display of the data. DMOIB component software does not display information. It provides an interface to gather the information from the dictionary in order to properly display based on how the dictionary defines it. Application software employs DMOIB software components to access the base dictionary and linked string table to facilitate the construction of intelligent applications. The link table can be used to gather the localized (language) representation of the member from the proper string table.

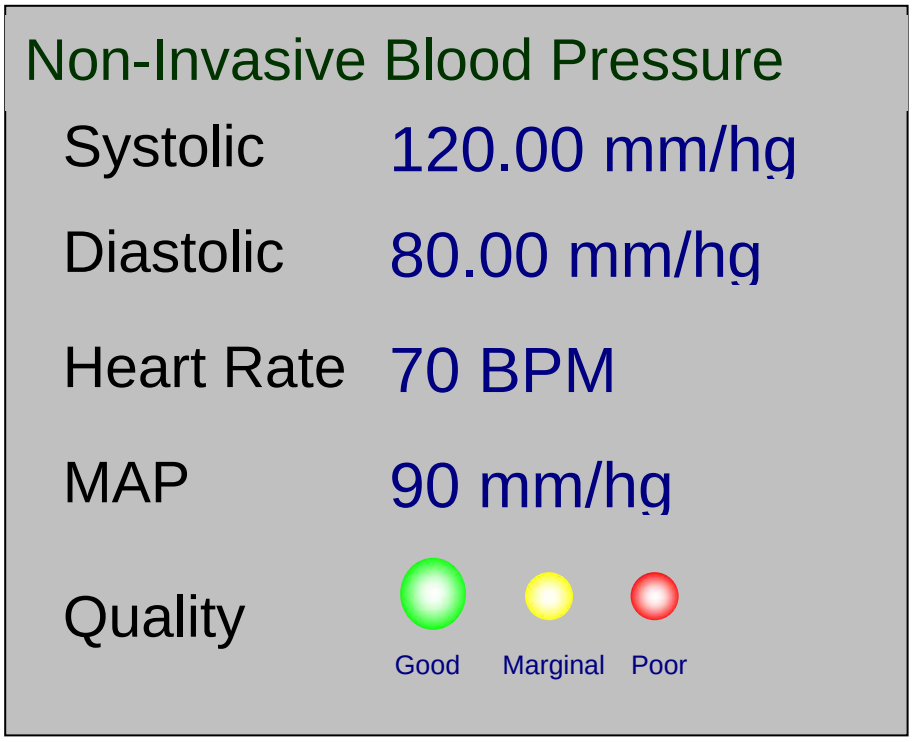
When intelligent applications display the MOIB information the attributes in the dictionary are applied to each value.

For Systolic the scale might be 0.01. The scale value is multiplied with the encoded data to produce properly formatted data. For example, the value in the systolic item is encoded in a 16bit numeric as 12,000(for example) with no precision (decimal). When multiplied by the scale factor 0.01, a value of 120.00 will result. Due to floating point encoding sizes and arithmetic differences between processors, precision data is normally sent as integer values requiring a scale conversion.

For diastolic the value might be 8,000. When multiplied by scale, e.g. 0.01, a value of 80.00 will result.

For MAP the value may be 9,000. When multiplied by scale, e.g. 0.01, the resultant value is 90.00.

Finally, the “Reading Quality” can be displayed by extracting the value of the final 8bit encoded data. The name attribute is displayed followed by the name corresponding to the number encoded in the 8bits of data. E.g. a value of 0 would mean the quality is Green, a value of 1 would be Yellow, and a value of 2 would be Red. Combining units provides the final display built from the information in the dictionary.



The dictionary allowed what you see above to be built by an application supporting the dictionary. The same process can be applied to any newly discovered information allowing for machine perception.

Additional Capabilities

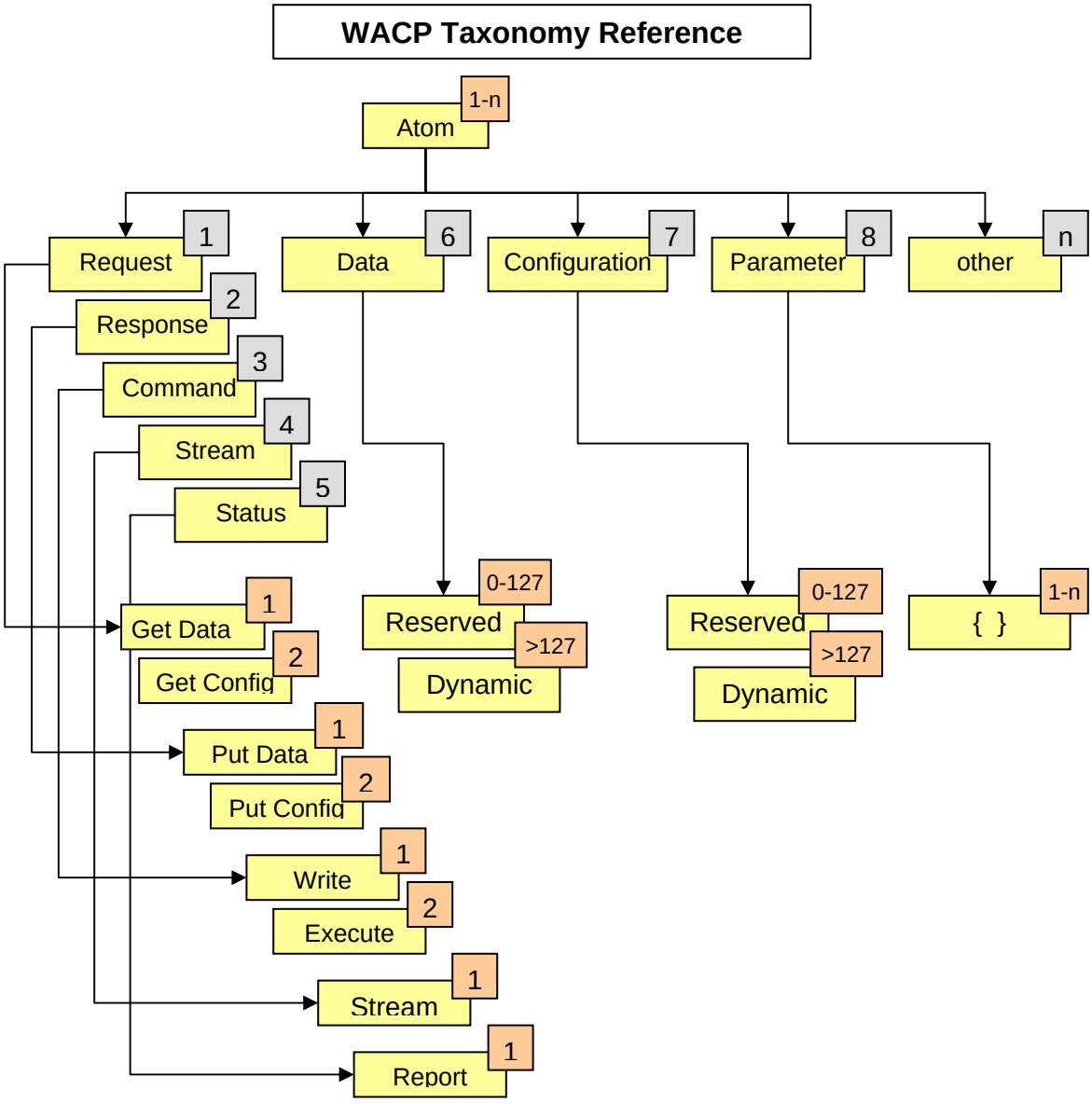
Before we explore the tools and product opportunities for WACP technology we need to explore some additional capabilities and uses. DMOIB is a useful technology for creating intelligent applications that adjust meeting the needs of adapting systems as we defined in our example above. There are machines unable to leverage self description due to processing and memory constraints. For these machines a traditional standard based structure is required. What exactly does this mean? Traditional communications systems reserve identification numbers representing well known structures of information (atoms) passed in communications. They are not expected to change. As an example a contract between network constituents might be the set {1,6,1}. This set represents an NIBP with four parameters each encoded in 16 bits in order of systolic, diastolic, heart rate, and MAP. The units are mm/hg and the scale is .01. The structure is predefined and all machines in the network agree to this and are programmed accordingly.

In our example taxonomy for blood pressure, the set {1,6,1} represents a numeric ID from the Blood pressure family. More precisely, it is a definition of a diagnostic Non Invasive Blood Pressure reading. Using the dictionary the receiver has a flexible mechanism for information extraction as defined by machine dictionary. If

the dictionary does not exist, the information has to be standardized and controlled in the traditional sense. For physiology components (hardware sensors) serving one function like temperature or blood pressure capture, the DMOIB approach is not employable due to performance constraints. In devices like these and others there are provisions supporting traditional reserved numbers in the taxonomy. This is done in each family by reserving numbers at the species level. The range is used to define fixed structures. Information types in this range are contractual. If they change there are rules for how as defined in MOIB.

In cases where DMOIB is not used the MOIB/MOMP specification still employs intelligent rule based dialog but not self description from the device. This does not limit changes to standard MOIB types. In simple terms an MOIB object of lesser version is **unable to receive** an MOIB object of greater version; however the reverse is true. An MOIB object of greater version contains the evolution history of the object. It knows when and how the object evolved and can adjust in processing based on what the atomic family defines. MOIB is backwards compatible.

The following diagram is the taxonomic reference for WACP expanded to include the reservation of standard types. The data species is split into two reservation sections. The first is standard based information governed by traditional paper contracts for information models. The other half is reserved for DMOIB. Machines developing taxonomies intending to deploy self description use the species values in a reserved set greater than 127.



The above taxonomy configuration employs the same model as data. Standardizing on configuration is difficult at best due to varying hardware platforms and algorithm options between machines. For configuration in WACP machines, dynamic machine dictionaries are highly recommended. This is beneficial in that a single configuration application can configure any WACP compliant machine by utilizing the dictionary.

Standard MOIB Information Evolution

Standard definitions are managed in a library of defined information (like paper based standards). Modifications are carefully managed. MOIB provides a mechanism for working with standardized information evolution. Individual items

in the MOIB library carry a mechanism for recognizing varying versions of MOIB information atoms. The “versions span” is a clue to the existence of a member in a received atom of information. MOIB implementation components and the machine dictionary (DMOIB) use a version spanning mechanism for each item to identify when and if it still exists for processing. In order to properly define this technology we look at the machine dictionary of our NIBP example. We have added an attribute to the dictionary for each member. This attribute defines each item’s history.

DICTIONARY TABLE - NIBP ATOM											
1	6	1	0x02	0x02	0x02	0x02					
1	6	1	Name	AKA	**V 1.0**						
1	6	1	Item 1	**1.0 -Cur**	Name	Units	Scale	AKA			
1	6	1	Item 2	**1.0 -Cur**	Name	Units	Scale	AKA			
1	6	1	Item 3	**1.0 -Cur**	Name	Units	Scale	AKA			
1	6	1	Item 4	**1.0 -Cur**	Name	Units	Scale	AKA			

If the NIBP atom is defined with our four original attributes and released for use, it would have a version reported as 1.0 as defined by the version attribute of the Atom. All encoded MOIB Atoms carry a version identifier. The version span attribute seen in each item (1-4) defines when the item existed in the Atom. For the first release each Version span attribute would be “Version 1.00 to Current” (1.0-Cur). If we add a new item following the 1.0 release the item is added and the atomic version is increased. The V-Span for the new item would be the “Version 2.0 to Current”.

The example below shows a new entry in the table. Item 5 has been added and the atomic version increased to 2.0. Item 5 was introduced in version 2.0 and therefore reports a history as 2.0-Cur. Received NIBP atoms use the received information reported version to determine what to process and what to ignore.

DICTIONARY TABLE - NIBP ATOM											
1	6	1	0x02	0x02	0x02	0x02	0x01				
1	6	1	Name	AKA	V 2.0						
1	6	1	Item 1	1.0 - Cur	Name	Units	Scale	AKA			
1	6	1	Item 2	1.0 - Cur	Name	Units	Scale	AKA			

1	6	1	Item 3	1.0 - Cur	Name	Units	Scale	AKA	
1	6	1	Item 4	1.0 - Cur	Name	Units	Scale	AKA	
1	6	1	Item 5	2.0 - Cur	Name	Units	Scale	AKA	

In processing standard types, MOIB first looks at the received version reported at the atomic level. It checks the received version against the current/working version for the MOIB atom. The version of the working atom must be equal to or greater than the version received in order to process the data. If it does not it will be missing important history about the items in the received object and be unable to extract the information properly. If the working MOIB atom is greater than or equal, it will use the received version of the atom against the version spans for each item to determine if the item is available. Receiving an object of version 2.0. the working MOIB extractor will use an electronic form of the table above to intelligently extract the data from what has been received. If a version 1.0 MOIB atom is received it will not process Item 5 because the dictionary table reports Item 5 as not existing until V 2.0.

In the next example we remove Item 4 and bump the version to 3.0. Notice that the dictionary does not remove the item reference. It can not remove history. It maintains the full historical evolution of the atom and therefore the ability to receive and process any version of the atom.

DICTIONARY TABLE - NIBP ATOM

1	6	1	0x02	0x02	0x02	0x02	0x01		
1	6	1	Name	AKA	V 3.0				
1	6	1	Item 1	1.0 - Cur	Name	Units	Scale	AKA	
1	6	1	Item 2	1.0 - Cur	Name	Units	Scale	AKA	
1	6	1	Item 3	1.0 - Cur	Name	Units	Scale	AKA	
1	6	1	Item 4	1.0 – 2.0	Name	Units	Scale	AKA	
1	6	1	Item 5	2.0 - Cur	Name	Units	Scale	AKA	

In processing a received version 3.0 of this atom, item 4 is skipped because the version span reports it's existence only from version 1.0 ending/removed at version 3.0. In this example of an evolving standard data type, we have a single dictionary managing an atom through 3 evolutions.

Mixing Standard MOIB with Dynamic MOIB

Full self describing machines do not require version spanning technology. Full self describing machines report the machine model in the data dictionary each time. A compute platform may cache the dictionary maintaining a running history of discovered devices. You might argue for machine based self description as a requirement so versioning is not required.

The reality is, lower power devices will not support full self description. The version spanning mechanism defined here is aimed at intelligence for managing standard information generally consistent and subject to only minor evolution if any at all. Standard types can be applied to the dynamic dictionary along with the pure dynamic types. This splits dynamic MOIB implementation into two information processors. The first is a pure discovery model in which the machine defines the information model. This is the pure dynamic model. In this model, since the machine communicates the model each time, the communicating application is nothing more than an interpreter of the machine model. In the second manifestation, built on static types, the communicating application maintains a memory of standard types and is constantly learning or replacing old version of MOIB definitions with newer updated models. Standard MOIB is useful in that it provides a base set of environmental attributes in a complex system that in many cases could define rules in the physical world. In our earlier example of the car that could drive on its own, in order for it to process an incident, it needs to determine action based on input from the physical world. In the example of the bowling ball thrown in front of a car, the car requires knowledge of the rules of gravity, color patterns, and texture to determine the projectile's potential for damage. Human cognition exhibits this ability. Physical or environmental attributes can be defined in a standard atomic model for a car's vision system. It can then be acted upon to taking the appropriate action. In this case, stopping or avoidance.

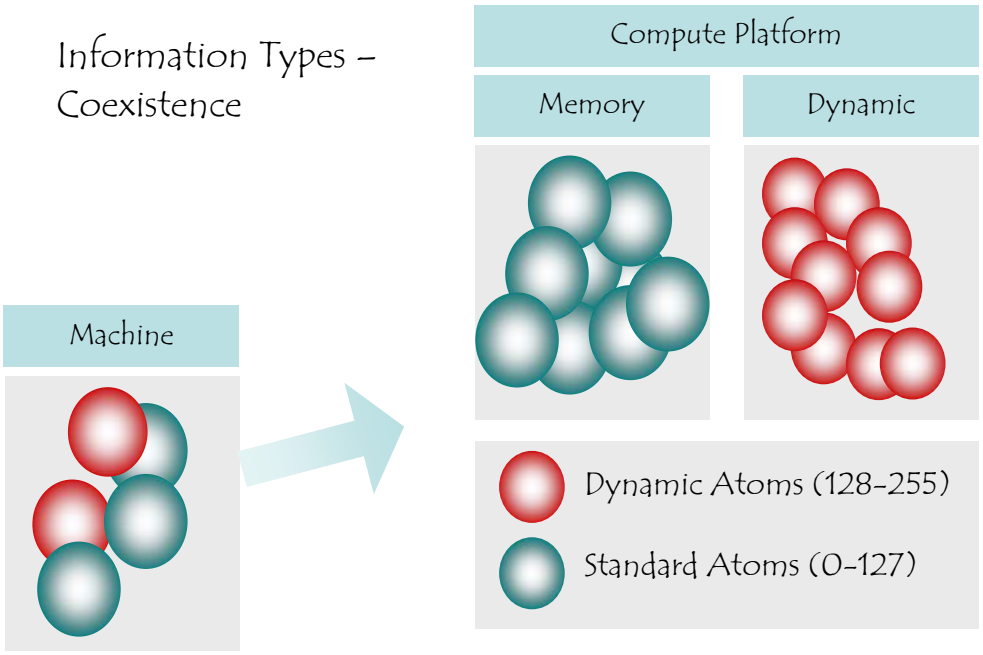
Dynamic MOIB compute applications must manage full self-describing machine dictionaries as well as global dictionaries of standard types. There are benefits and drawbacks to both. Fully dynamic machines are free to define information modes complimentary to the device. It is likely the model will be unique to the machine. This requires a DMOIB compute platform to manage many full dictionaries from one machine to the next. The standard MOIB approach is universal across machines and easier to manage in an application that understands common information models across many machines. A environment supporting both is desirable. Why… complex environments are constantly changing and the world is wrought with complexity.

For standard based MOIB definitions a compute platform must keep up-to-date dictionaries in order to interoperate and handle the evolution of atomic information models reserved in the standard space. DMOIB Applications must manage and store the standard types like memory in the human mind. The DMOIB application might also have to manage the dynamic information types reported by devices. The standard types are consistent across all devices. An

MOIB application “brain” would be a conglomeration of newly discovered atoms along with atoms evolving over time. The Intelligence mechanism would check the atomic versions for a standard type like our NIBP atom. If it encounters a version of greater value for this standard type it would commit it to memory learning as machines it encounters evolve the data.

A model employing both is able to leverage interoperability regardless of the capability of machine. DMOIB implementation dictionaries can manage/map universal standards based information as well as full described machines in DMOIB dictionary form. This enables the application to work with self describing and non describing devices.

The following diagram below is a representation of a compute platform exhibiting the two uses of MOIB. Standard based MOIB are considered/defined as applied memory. While dynamic can be considered disposable but it may also be cached.



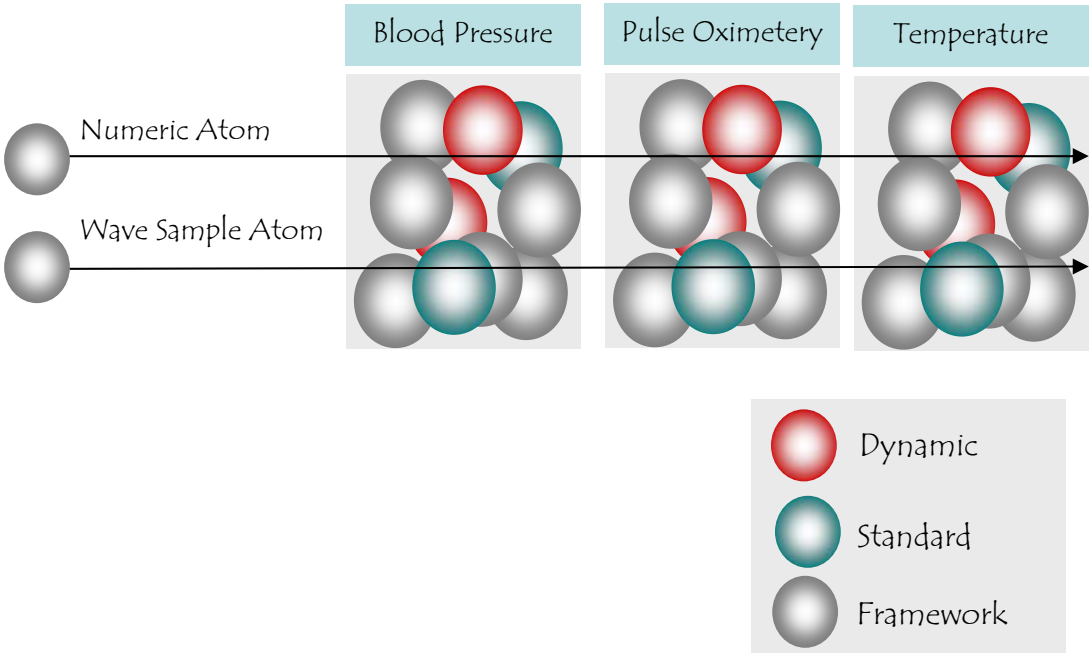
The diagram above defines a likely scenario for a machine interacting with a compute platform. The machine in this example has both standard and dynamic atoms in it’s machine dictionary. The compute platform exhibits a dual role as we defined earlier. It contains an evolving memory base of known physical world attributes or behaviors. Like the machine, humans also employ this trait as can be seen in the selective nature we exhibit in learning (or as my wife might call it… selective hearing). Standard types as we defined earlier are the foundation or building blocks and may represent a foundation for decision making. This does not completely rule out dynamic atoms in a decision making process. Dynamic

atoms contain an AKA (also-known-as) in the dictionary. AKAs are taken from a global list representing a widely recognized nomenclature. For example a machine may define a heart rate as “HR” or spell it out as “Heart Rate”. Attempting to process dynamic data in a disease based trending/clinical engine would be problematic in that the text ID of heart rate is defined differently across machines. The AKA is an agreed upon number representing a clinically recognized information part. Humans exhibit the ability to look at information sets and extract well known information types putting together clues on the intent of a combined set of data. Imagine a storage facility containing dynamic atoms gathered from thousands of machines from hundreds of manufacturers. The AKA identifier would allow an intelligent process to search the data store to find all of a particular set of information and process the data for disease trends (in a medical application for example).

Standard Framework Types:

Applying what we have learned about standard and dynamic MOIB we will explore the standard type further in a specific application model expanding our atomic universe to include a Standard ***“Framework”*** type.

The MOIB library supports defining a standard-framework type. The traditional development of atomic physiologies in families is one method employed by the taxonomic model in WACP. In our examples we have been focusing on the atomic model of the Blood Pressure family. In this family we define information and configuration for the atomic family. For a medical manufacturer the definition of diagnostic data mainly focuses on physiology. MOIB is a taxonomic mechanism for information modeling and identification and is not specific to just diagnostic data modeling at a physiologic level. In diagnostic function, atomic families of physiology like blood pressure, temperature, height, weight, SpO2, ECG, etc are defined. These atoms have data definitions, configurations, and messages. There are other taxonomies that are not diagnostic atoms in the same sense. These categorical atoms are framework based and extended across physiologies representing a shared information model with a common consistent information set. The taxonomic model is capable of defining any categorical information framework. Categorical atoms can be mapped across all defined physiologies and represent an information category like a numeric or wave sample. Examining blood pressure or SpO2 we can classify numeric types with common attributes across the two physiologies The diagram below defines how the taxonomy intersects families for this type of definition. Each atomic family defines itself vertically. Framework taxonomies intersect each atomic family defining a common framework for representing categorical types of information. These framework types share information attributes as well as configuration attributes.



Framework atoms do not violate the taxonomic model. Framework definition is beneficial for a number of reasons. Management of an individual numeric taxonomy sharing consistent traits in format and configuration in a single atom adheres to the classification mechanism in the science of taxonomy. Separating and defining numeric or wave in vertical physiologies or families creates a dispersed definition. This opens up the opportunity for errors introduced in the definition library when updates are not executed across all families containing standard framework types. It also produces a deficiency in configuration taxonomy.

To help better understand the challenge of developing framework definitions in the vertical physiologies we attempt the definition. The development of the numeric in the vertical families might take the following form. Notice numeric configuration is not clear in the taxonomic model.

Information Taxonomy NIBP

Aggregated data

FmNIBP::GnData::SpStandardReading

Numeric Data

FmNIBP::GnNumeric::SpSystolic

FmNIBP::GnNumeric::SpDiastolic

FmNIBP::GnNumeric::SpHeartRate

FmNIBP::GnNumeric::SpMeanArterialPressure

Configuration

FmNIBP::GnConfguration::SpStandardReadingConfig

FmNIBP::GnConfguration::SpSystolic ***(systolic “numeric” configuration?)***

FmNIBP::GnConfguration::SpDiastolic ***(Diastolic “numeric” configuration?)***

FmNIBP::GnConfguration::SpHeartRate ***(Heart Rate “numeric” configuration?)***

FmNIBP::GnConfguration::SpMeanArterialPressure ***(MAP numeric configuration?)***

There is a taxonomic anomaly in applying a numeric in vertical families. Numeric configuration is consistent across each numeric. In the model above the configuration of the numeric for each data numeric defined is not clear. Even if you applied a model whereby you defined the following:

FmNIBP::GnConfguration::SpNumeric

Or

FmNIBP::GnConfguration::SpNumericSystolic

There is still no way of identifying the numeric species owning a configuration in the taxonomy. In the second there is a clear overloading of the species violating the taxonomic process and adding complexity to species processing.

Each numeric data type requires it’s own configuration attributes, for example, limits. The definition in vertical physiologic families is not capable of identifying the taxonomy as a configuration type for a specific species of numeric type. Any attempt made tends to violate the taxonomic structure for family genus and species.

In attempting to mitigate issues like this we turn to taxonomy. The preferred taxonomic model is detailed below. Notice the specificity in both data and configuration and the reuse of species to identify both the data and configuration.

Information Taxonomy Numeric

Data

FmNumeric::GnData::SpSystolic

FmNumeric::GnData::SpDiastolic

FmNumeric::GnData::SpHeartRate

FmNumeric::GnData::SpMeanArterialPressure

FmNumeric::GnData::SpTemperature

FmNumeric::GnData::SpSpO2

FmNumeric::GnData::Weight

Etc…

Configuration

FmNumeric::GnConfguration::SpSystolic

FmNumeric::GnConfguration::SpDiastolic

FmNumeric::GnConfguration::SpHeartRate

FmNumeric::GnConfguration::SpMeanArterialPressure

FmNumeric:: GnConfguration::SpTemperature

FmNumeric:: GnConfguration::SpSpO2

FmNumeric:: GnConfguration::Weight

Etc…

The taxonomy in this model clearly defines the atomic members in a single family. Using a visual mechanism the numeric family identification is defined as follows.

Family Numeric – Genus Data – Species N

x 6 1 = Systolic Numeric – Species 1

x 6 2 = Diastolic Numeric – Species 2

x 6 3 = Hear Rate Numeric – Species 3

x 6 4 = Mean Arterial Pressure – Species 4

x 6 5 = Temperature - Species 5

x 6 6 = SpO2 – Species 6

x 6 7 = Weight – Species 7

Family Numeric – Genus Configuration – Species N

x 7 1 = Systolic Numeric – Species 1

x 7 2 = Diastolic Numeric – Species 2

x 7 3 = Hear Rate Numeric – Species 3

x 7 4 = Mean Arterial Pressure – Species 4

x	7	5	= Temperature - Species 5
x	7	6	= SpO2 – Species 6
x	7	7	= Weight – Species 7

In both configuration and data the species is the same and the taxonomy clearly defines what is represented in the atomic member. It is clearly expandable and can be maintained in a single atomic family.

Framework definition is advantageous for other reasons. A numeric configuration will carry the configuration limits for the numeric. Each numeric has the potential to employ an upper and lower limit. Because the configuration for any numeric is consistent, the framework atom can support a universal user interface that can handle any numeric. Before we look at that potential we define the atoms for configuration and data in the numeric family.

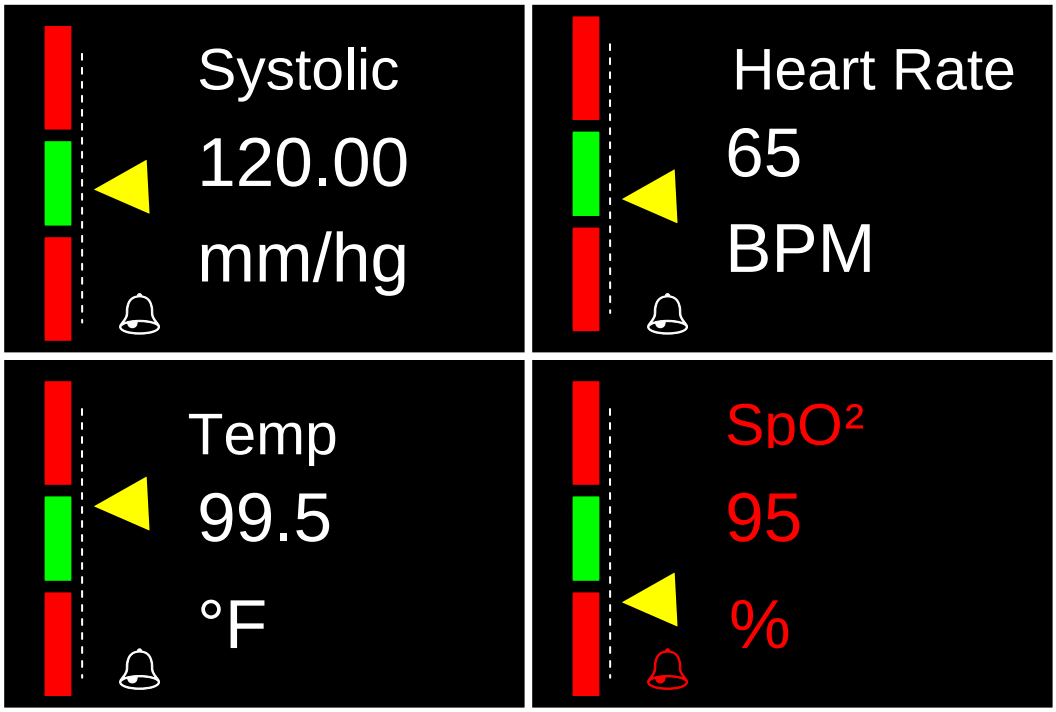
DICTIONARY TABLE – Numeric Data Atom								
x	6	n	0x04					
x	6	n	Name	AKA	V 1.0			
x	6	n	Numer	1.0 - Cur	Name	Units	Scale	AKA

DICTIONARY TABLE – Numeric Configuration Atom								
x	6	n	0x04	0x04				
x	6	n	Name	AKA	V 1.0			
x	6	n	UpLim	1.0 - Cur	Name	Units	Scale	
x	6	n	LwLim	1.0 - Cur	Name	Units	Scale	

The value n represents any species in the dictionary defined earlier. E.g. systolic=1, diastolic=2, Heart Rate=3, etc. The species value is common to both data and configuration.

The set {x,6,1} and {x,7,1} represents the data and configuration for a systolic reading. The ‘X’ is a number representing the assigned numeric value for the “Numeric” atomic family.

Taking the atomic definition a UI can be constructed to support the taxonomic model. An example is provided below.

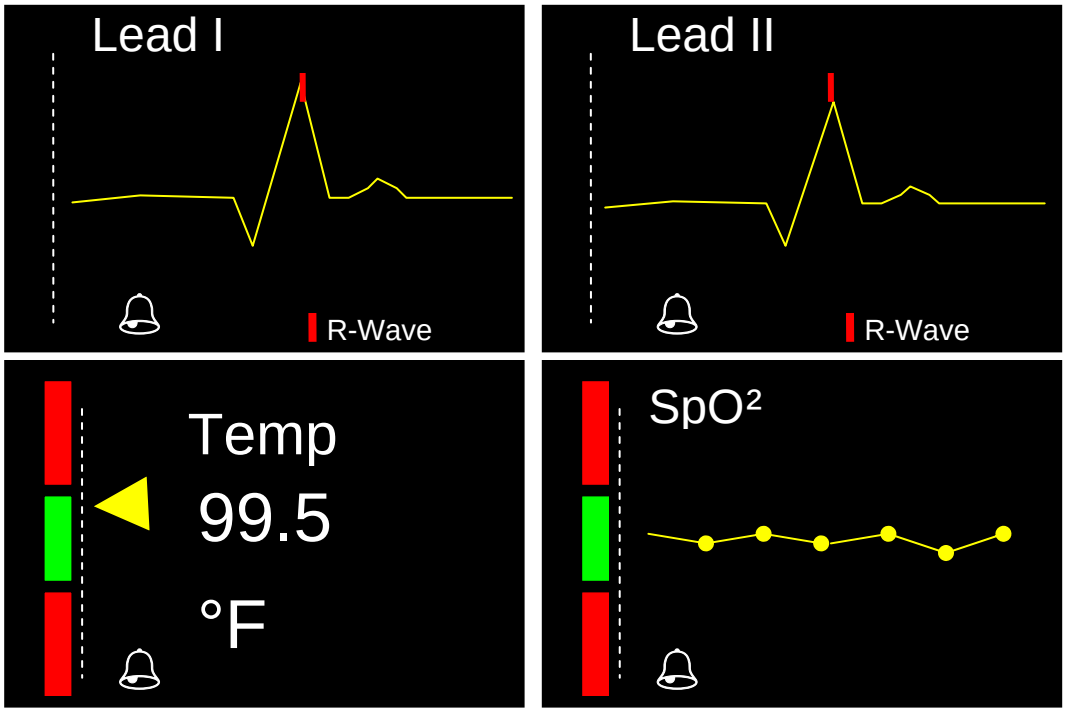


Examining the UI above it is easy to see how a single atomic framework family can be used to drive a smart user interface. The UI in this example uses the dictionary and reported configuration to handle any numeric encountered.

A Dynamic MOIB platform application would receive the dictionary from a monitoring device. The dictionary defines the numeric types supported by the device and the human readable strings defining each numeric species. The compute platform creates a single UI for each reported numeric. It sets the attributes of the UI to display the proper textual representation, units, and applies the proper scaling mechanism so the data is properly represented.

In order for the gauge to fully function it would retrieve the device configuration for each numeric and apply it for each numeric. The gauge would update as the reading changes. The slider gauges on the left represent the current limit status. The green in the middle represents a reading within limit and red out of limit. If an out of limit condition is encountered the display changes the bell and the text to red for an enhanced visual cue.

In addition to the numeric framework types there is a wave sample used to define wave information. In the diagram below the same UI concept is used to create a framework based UI component for wave samples.



Standard Framework types provide a powerful mechanism for monitoring applications. A smart monitoring application can handle any machine discovered by utilizing standard smart UI interfaces linked to standard framework types.

To recap, MOIB supports three atomic types as defined below.

Standard Atoms are those objects in a particular family that are classified as standard information and are fixed in format. For devices that are unable to support self description due to overhead, these types provide set members of an atomic family to interoperate without a data map. Standard data types are identified by a CLSID with a species of 0x00h-0x7fh.

Framework Atoms are objects representing a classification type of information like a numeric or a waveform. They, like standard types, are controlled in a contractual manner via the MOIB information library. Standard framework data types share the allocation space with standard atoms.

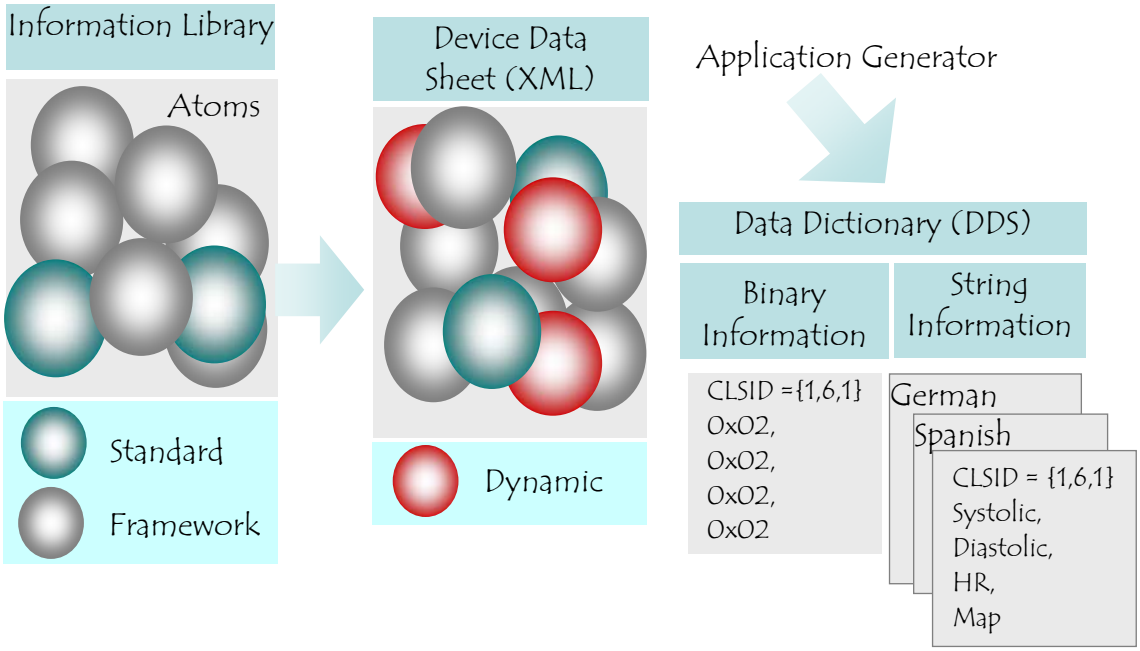
Dynamic Atoms are developed against the self describing framework by the device at the time the device model is built. This dynamic part can leverage any nomenclature available (as well as employ the AKA). Dynamic types are reliant on self description. E.g. the map and display names are required in order to extract, display and format the data in an extensible form for consumption by external systems. Dynamic data types are identified by a CLSID with a species of 0x80H to 0xFFH.

Tools and Extensibility

The design goal of WACP is a binary protocol. It serves this capability very well. In working with the taxonomic approach it became clear the taxonomy could very easily be represented in XML. Early in the development of WACP, XML was added as an option for serialization and as the modeling tool for creating and maintaining a library of atoms leveraged as new machines models are constructed. A single atom of NIBP is represented in XML below. It represents the taxonomic structure defined in a previous section of this document.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
- <FAMILY name="FmNIBP">
    <!-- ================================== -->
    <!-- GnDATA  -->
    <!-- ================================== -->
  - <DEFINITION class="CNIBPDStd" abrv="CNBPDSTD" version="100" family="FmNIBP" genus="GnDATA"
      species="SpSTANDARD">
    - <MEMBERS>
        <!--  Version 001   -->
        <MEMBER type="int16" name="Systolic" minversion="100" maxversion="CURRENT_VERSION"
          scale="0.01" units="mm/Hg" AKA="10024" comment="Systolic BP {in 0.01 mmHg units}" />
        <MEMBER type="int16" name="Diastolic" minversion="100" maxversion="CURRENT_VERSION"
          scale="0.01" units="mm/Hg" AKA="10025" comment="Diastolic BP {in 0.01 mmHg units}" />
        <MEMBER type="int16" name="MAP" minversion="100" maxversion="CURRENT_VERSION"
          scale="0.01" units="mm/Hg" AKA="10026" comment="Mean Arterial Pressre {in 0.01 mmHg
          units}" />
        <MEMBER type="uint16" name="HR" minversion="100" maxversion="CURRENT_VERSION"
          units="BPM" AKA="10050" comment="NIBP Heart Rate (in BPM (Beats/Minute) units}" />
      </MEMBERS>
    - <SPECIES_TABLE>
        <SPECIES name="SpSTANDARD" defaultvalue="0" />
      </SPECIES_TABLE>
    </DEFINITION>
    <!-- ================================== -->
    <!-- Messages                                                    -->
    <!-- ================================== -->
  - <MESSAGES>
      <MESSAGE family="FmNIBP" genus="GnREQUEST" species="SpGET_DATA" defaultvalue="128"
        version="100" />
      <MESSAGE family="FmNIBP" genus="GnREQUEST" species="SpGET_CONFIG" version="100" />
      <MESSAGE family="FmNIBP" genus="GnRESPONSE" species="SpPUT_DATA" version="100" />
      <MESSAGE family="FmNIBP" genus="GnRESPONSE" species="SpPUT_CONFIG" version="100" />
      <MESSAGE family="FmNIBP" genus="GnCOMMAND" species="SpWRITE" version="100" />
      <MESSAGE family="FmNIBP" genus="GnSTATUS" species="SpREPORT" version="100" />
      <MESSAGE family="FmNIBP" genus="GnSTREAM" species="SpSTREAM" version="100" />
      <MESSAGE family="FmNIBP" genus="GnCOMMAND" species="SpEXECUTE" version="100" />
    </MESSAGES>
  </FAMILY>
```

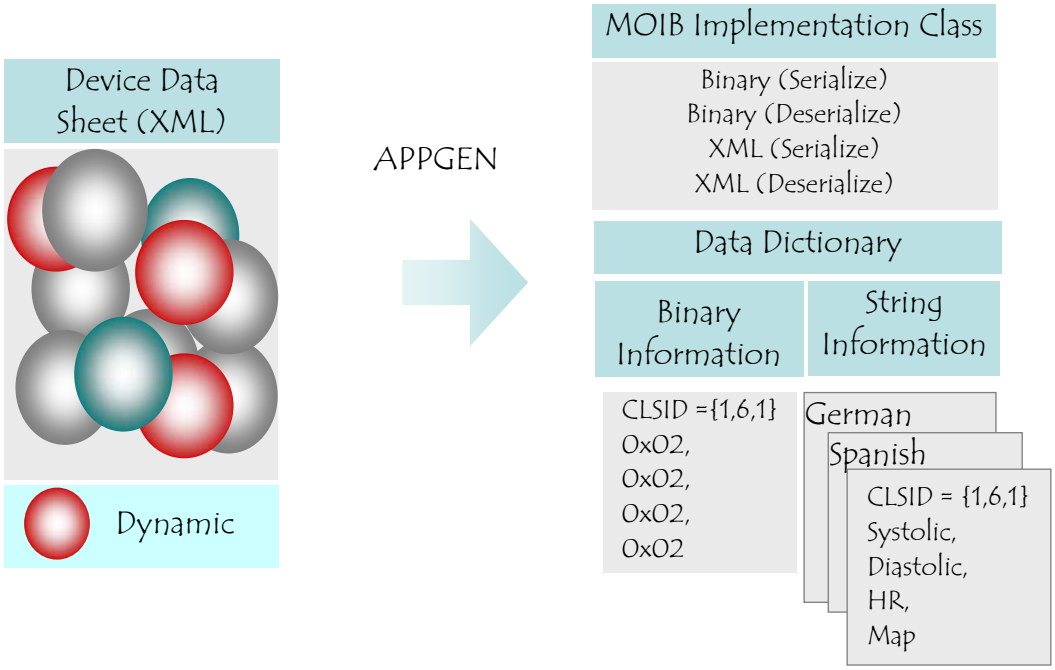
Project teams building a new machine use XML and MOIB rules to define the semantic model of a device. They take from the MOIB library which contains the currently defined universe of standard and framework atoms. The diagram below defines this process. The definition process starts with the MOIB library. Standard and framework atoms are extracted from the library starting the construction of the Device Data Sheet (DDS). At the second stage the device may define dynamic atoms. Once this is complete the XML is converted to software via the Application Generator compiler.



Once the process above is complete the software components can be integrated to the device.

The components created by the process above provide the facilities for information implementation (MOIB) and communications (MOMP). Serialization is the process taking an instance of software structure or class and turning the attributes/members of the class into transferable data encoding. MOIB objects and MOMP message objects support serializing information to binary and XML for transfer or use by applications.

Serialization to XML is advantageous in that XML has industry wide accepted tools for transforming the data into other formats. MOIB implementation can act as a protocol bridge. It can communicate with binary systems as well as bridge the data to information systems dealing with text based or higher level protocols like HL7. Serializing to XML and applying it to transform engines allows the MOIB based binary networks to bridge to other protocols like HL7 and other forms of XML specification. The following diagram expands on the previous defining this capability. Each atom instance in a machine is capable of serialization to a either XML or binary. This is inherent to all MOIB components produced by the Application Generator (APPGEN)



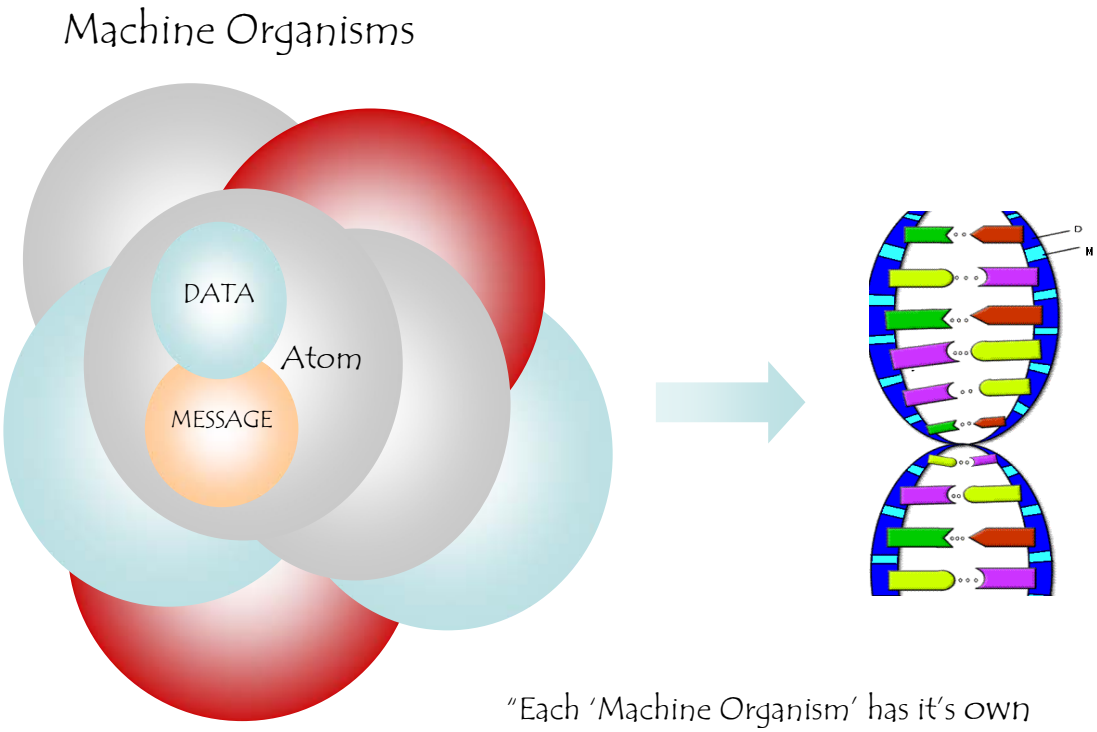
MOIB implementation defined in the diagram above defines Binary and XML serialization and deserialization. It can receive (deserialized) XML or binary as well as serialize it to both formats for transfer.

A note about implementation… in the diagram above MOIB implementation as produced by APPGEN can utilize the data dictionary for information extraction and display. APPGEN produces an implementation not reliant on the dictionary. In this form MOIB components have the structures of information built in and therefore do not utilize the dictionary.

XML represents the full taxonomic model and can be used as input to APPGEN compiler creating the complete software implementation of the WACP communication stack and the implementation of the either static MOIB objects or the Dynamic MOIB data dictionary. The data dictionary is a binary form of the XML and is used as the mechanism to decode and encode data for a specific atomic member. Like a recipe the data dictionary is used to decipher the makeup of an atomic family member from a raw binary stream or create one for populating. As we defined earlier an atomic member maybe a configuration set, information or data set, or a parameter set.

XML is currently used as the modeling tool for device information and messaging models as seen in the NIBP example above. XML entities like the NIBP sample are combined with others to form device data sheets. Device data sheets form the atomic representation of the device. These device data sheets (DDS) can be applied to the toolset to generate the binary form of XML we call the data dictionary. The dictionary is linked to as many language specific string tables as needed. The link and string tables can be documented in XML and applied to the Application Generator tool (APPGEN) to provide implementation in platform independent string software.

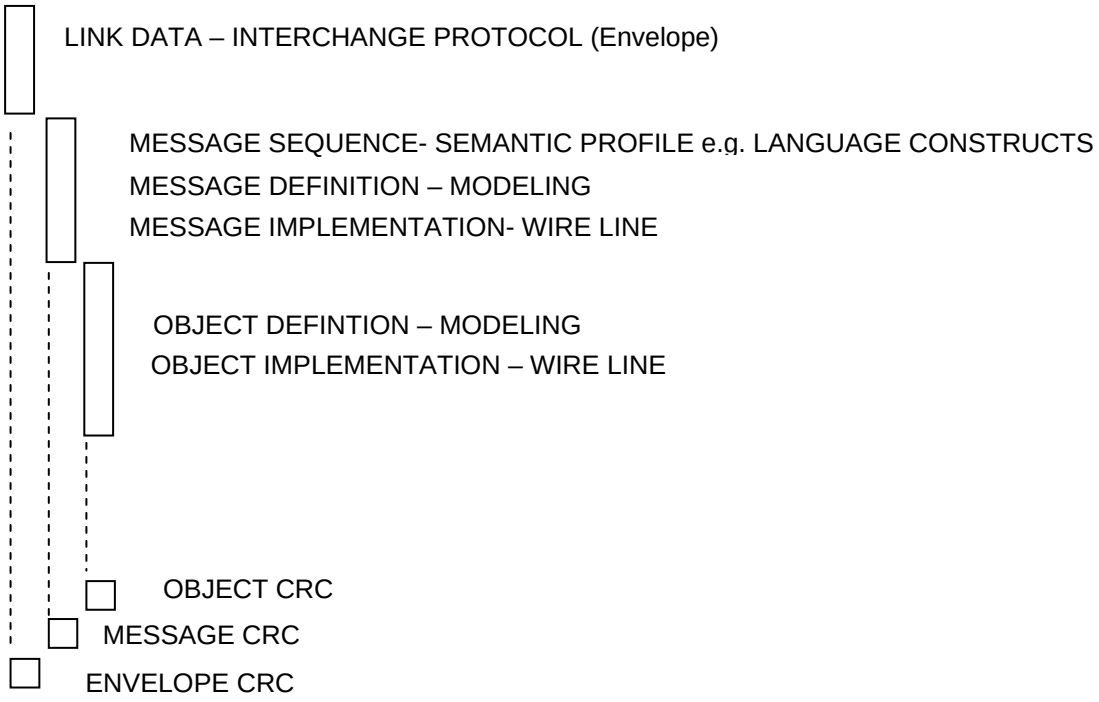
APPGEN provides a documentation mechanism for creating very complex models of information along with the messages for moving the information. Remember that message constructs are agreed upon for all machines and they must be adhered to so intelligent dialog based on machine grammatical rules can provide an open dialog for all machines. WACP technology defines each machine as having it's own fingerprint made up of multiple atomic families of information. Some of those items are base (standard and framework) items required in all machines while others are unique to the device. If the structure is defined as decipherable by constituents of a system, a true plug and play capability exists. This would allow devices to define information in models that could be learned as they are introduced on a network.



"Each 'Machine Organism' has it's own fingerprint made up of multiple atomic families of information. Dynamic MOIB provides a readable structure to discover changing/evolving information

Implementation

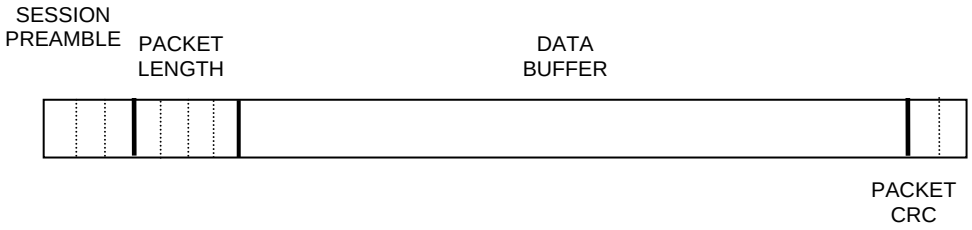
In the models defined at the beginning of this primer, we explored how WACP is a communication system similar to real world models. In this section we look at how this is applied in a simple implementation exercise (form more in depth definition see design documentation). To begin, the following diagram represents the base protocol structure as it exists in wire line implementation (byte streams of information). Each was previously defined in relation to our models. To recap; the link data is the envelope carrying payload. The message is the predicate or verb and the object the subject. Combined message and object represent the information modeled after language. The diagram below is what you will see in electronic form on the communications transport and is the form all WACP based dialog takes when communicated on a transport.



Envelope

Wire Line Specification

Envelope protocol used to carry conversation. WACP uses a simple preamble envelope CTL'W', CTL'A', CTL'L' with a size and a 16bit CRC for data payload integrity.



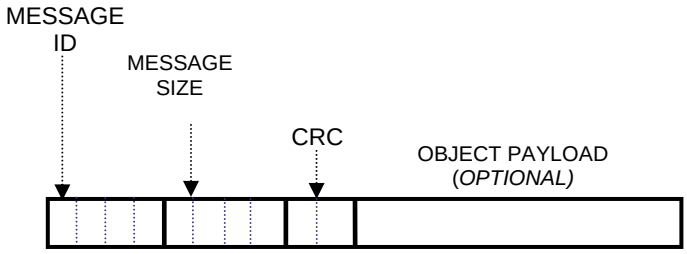
Information

Wire Line Specification - Message

The Family, Genus, and Species methodology employed by WACP has genus provisions for the definition of message genus types along with the data with in the context of a single 32 bit number allocated to a physiologic family.

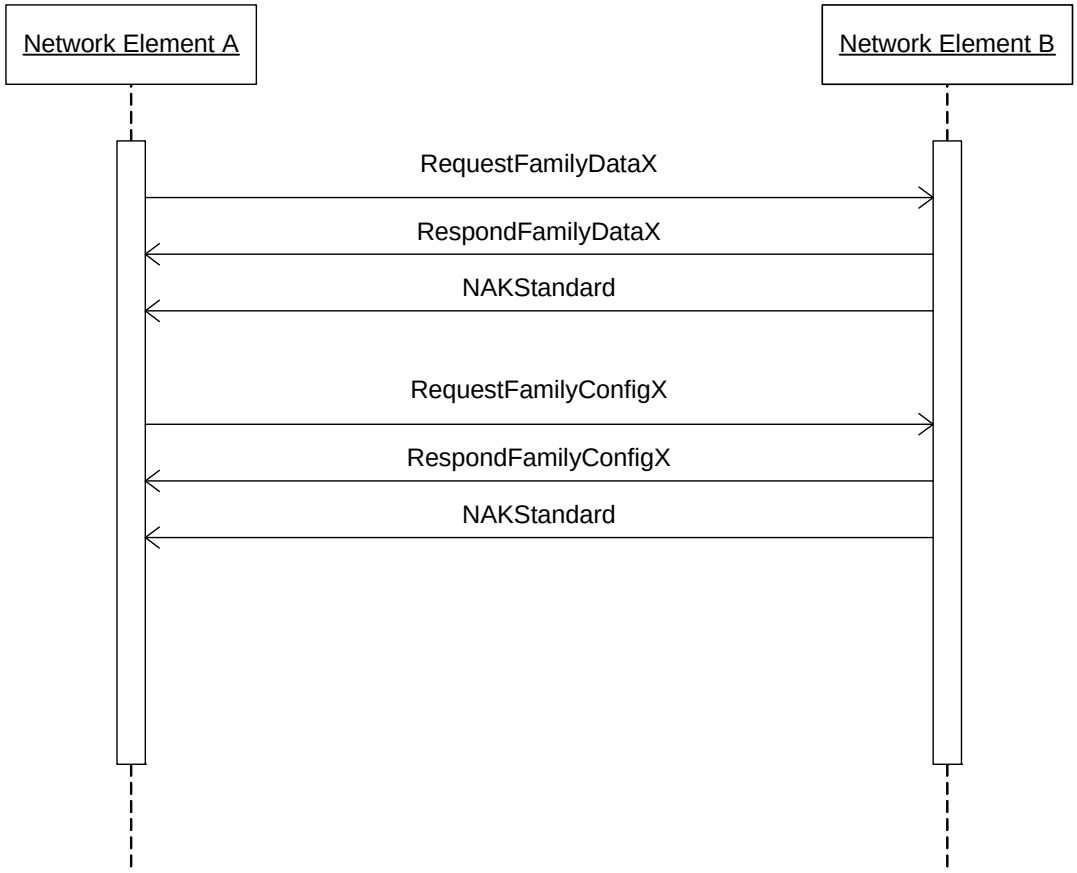
The wire line will require at a minimum:

- Message Op-code: identifying the message context.
- Message Size: the amount of data potentially found in the message
- Message Integrity: A 16 CRC or similar method of message integrity insurance



Base message signatures are defined here as well. These base dialog rules are simple constructs each physiologic family is required to support for data management/dialog across machines. The intent is for machines to be trained on message signatures for open dialog. Classified information atoms are required to support the basic message management function. i.e. all machines must support the request/response signature, status report message signature, and write command signatures. To reiterate this will allow simple management of any WACP device's data and configuration. Like humans trained in language, machines in interoperable systems must also be capable of supporting the rules of intelligent dialog. The sequence below is an example of a standard message signature for WACP devices.

Request Response Genus/Information Management

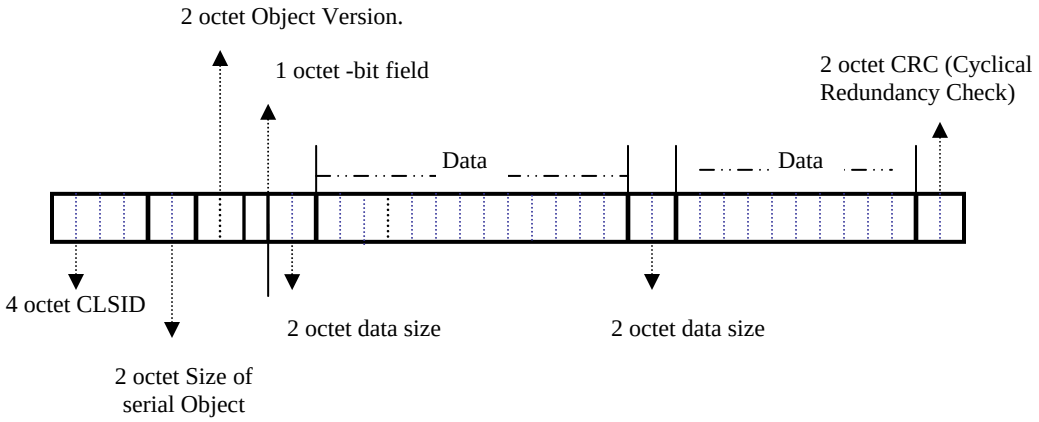


Wire Line Specification - Object

The Family, Genus, and Species mechanism employed by WACP has genus provisions for the definition of the data with in the context of a single 32 bit Number allocated to a physiologic family. See the examples provided.

The wire line will require at a minimum:

- Data Op-code: identifying the data carried
- Data Size: the amount of payload found in the data container
- Data Integrity: A 16 CRC for data integrity assurance.
- Version
- Compression type



Dynamic MOIB and Handshaking Specifications

These specifications were identified as support protocols. They function to detail how machines connect and share information taxonomy via self description. It is also important to remember the data dictionary and Dynamic MOIB is optional. Standard MOIB implementation may be used in place of the DMOIB implementation (DMOIB relies on the data dictionary).

Connection Negotiation Specification - Handshaking

A Specification for the connection handshake and transfer of the self described map is required. There are a number of ways to do this. A simple “Hi”, I’m a WACP device and here is my data dictionary (or a GUID to link to my map) would constitute the minimal set of requirements. The sequence needs to be standardized. The data dictionary may come from the device or be linked to the device via a GUID linking the device to a device’s cached map.

Data Dictionary Map

The following diagram is a simple text based mapping structure supporting adaptive self described technology for simply extracting data from a binary stream. The string data that follows is optional string data generated at the time of creation. It can be used in place of language based string tables found in the next section.

Sample Data Map

```
<DATAMAP>
65824|0:0x02h|1:0x02h;
---Optional---
65824:”CardioVascularPulseRateDynPleth”
65824|0:”PulseRatePleth”,U:”Beats Per Minute”,S:”0.01”,A:Pulse
65824|2:”Quality”
65824|2-0:”Green”
65824|2-1:”Yellow”
65824|2-2:”Red”
```

```
</PHDDATAMAP>
```

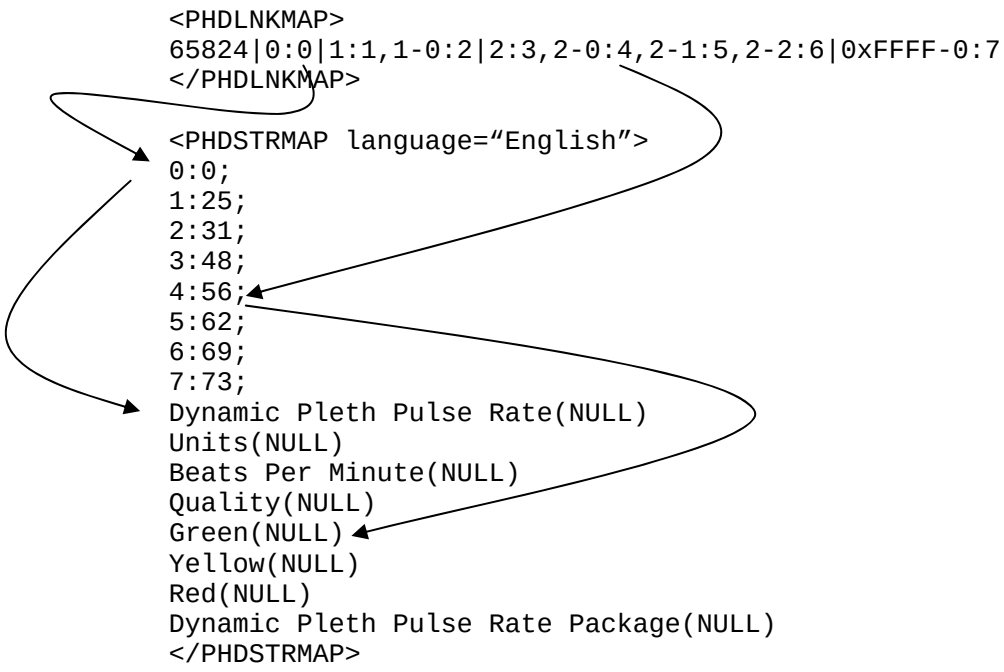
Using the map above a received ClassID can be broken down into its constituent parts for processing. Notice that Pulse Rate with class value 65824 has three preceding type specifiers (0x02h = uint16) because it has extra variables for units and quality of the reading. Additionally the map has primitive strings from the original definition that can be used if needed, however as it states these are optional.

String Table

The base data dictionary defined earlier supports a string. However another approach is to define strings via a link so string can be shared in a device between the dynamic self description and the display needs of the device. This eliminates unnecessary repetition of strings used in self description and on the device. The English string map can be found below along with the link table that is used to determine the string for each member.

In the link table the members are separated by the '|' symbol. "0:0" is the link for the first item. The value following the ':' in the link entry "0:**0**" is zero. This represents an array location in the actual string table header that houses the offset into the string table that represents the item's text representation.

Sample String Map



```
<PHDLNKMAP>
65824|0:0|1:1,1-0:2|2:3,2-0:4,2-1:5,2-2:6|0xFFFF-0:7
</PHDLNKMAP>

<PHDSTRMAP language="English">
0:0;
1:25;
2:31;
3:48;
4:56;
5:62;
6:69;
7:73;
Dynamic Pleth Pulse Rate(NULL)
Units(NULL)
Beats Per Minute(NULL)
Quality(NULL)
Green(NULL)
Yellow(NULL)
Red(NULL)
Dynamic Pleth Pulse Rate Package(NULL)
</PHDSTRMAP>
```

COMBINED DATA AND STRING MAP

The full adaptive solution will require the use of both the data map and the string map.

Combining Maps:

```
<DATAMAP>
65824|0:0x02h|1:0x02h|2:0x02h;
---Optional---
65824:”CardioVascularPulseRateDynPleth”
65824|0:”PulseRatePleth”
65824|1:”Units”
65824|1-0:”BeatsPerMinute”
65824|2:”Quality”
65824|2-0:”Green”
65824|2-1:”Yellow”
65824|2-2:”Red”
</PHDDATAMAP>


<LNKMAP>
65824|0:0|1:1,1-0:2|2:3,2-0:4,2-1:5,2-2:6|0xFFFF-0:7
</PHDLNKMAP>
<STRMAP language=“English”>
0:0;
1:25;
2:31;
3:48;
4:56;
5:62;
6:69;
7:73;
Dynamic Pleth Pulse Rate(NULL)
Units(NULL)
Beats Per Minute(NULL)
Quality(NULL)
Green(NULL)
Yellow(NULL)
Red(NULL)
Dynamic Pleth Pulse Rate Package(NULL)
</PHDSTRMAP>
```

XML Hook

Using the string table it is very easy to find the hooks that allow for XML output. The string map provides a bridge to XML. XML will fall out of the combination of both maps. Using the special extended object the string table can be used to produce the data in XML.

```
<Dynamic Pleth Pulse Rate Package >
    <members>
            <member name=”Pulse Rate” value=”60” type=”unit16”/>
            <member name=”Units” value=”Beats Per Minute”
type=”unit16/>
            <member name=”Signal Quality” value=”Green”
Type=”unit16/>
    </members>
</ Dynamic Pleth Pulse Rate Package >
```

This added benefit will allow for much easier integration of information from the binary network to text based networks like HL7 or HTML. Once in XML transforms can be used to change the data to many other formats.