

Standards For Transmission and Distribution System Integration: Towards “plug and play” of Utility Applications

Summary

This paper focuses on the emerging utility standards in support of the creation of a plug and play application and data integration environment. In particular, the paper will focus on the standards created within IEC Technical Committee (TC) Working Group (WG) 13 and 14 as well as how these standards can be used to minimize the effort required to implement application integration and data warehousing systems.

Introduction

The current economic climate and market initiatives require utilities to perform more efficiently and in more flexible ways. The dynamic nature of today’s environment means that a utility must be able to build an integration infrastructure for operational application integration and data warehousing quickly to provide a base for knowledgeable and adaptable business models. A commonly accepted way to achieve a flexible software infrastructure is via the use of plug and play components. Plug and play means that best of breed applications can be installed, integrated and upgraded or swapped more simply and at a lower cost.

To create a plug and play environment for utility operational integration, several elements must be agreed upon. First, applications must all employ a common meaning for the information they exchange. Second, applications must all employ a common set of mechanisms by which they connect to each other and exchange information. The first requirement addresses “what” data is exposed and the second addresses “how” data is exposed. While establishing agreement on both of these issues is required to achieve complete plug and play, complete agreement on all aspects of these issues is not possible. This does not mean that the amount of effort required to perform integration cannot be minimized via the use of standards. Rather, this means that because of the heterogeneous nature of legacy applications, complete interchangeability of applications cannot be realized. This paper lays out the extent to which plug and play can be achieved as well as the standardized technology required to do so.

Common Models

In order to precisely describe the meaning of a set of terms, engineers often create an information model. An information model describes a collection of related real world objects. The model provides unique names and definitions to each object and describes the relationship between them. The EPRI/IEC Common Information Model (CIM) describes data typically used in a utility’s operational systems. This includes data in an EMS or SCADA system, as well as data found in DMS, work, and asset management systems. More recently, the CIM is being extended to include transmission reservation and energy scheduling information. In general, the benefit of creating an information model include:

- Models give context to data improving understanding and productivity.
- Models enable automation of setup and maintenance tasks.

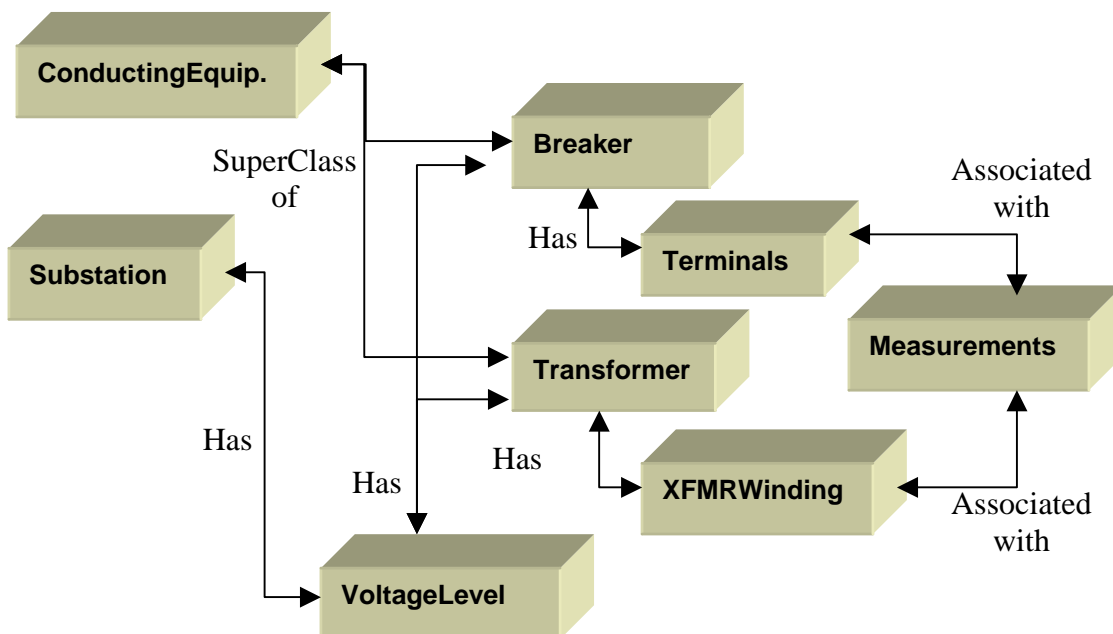
The CIM was originally developed as part of the EPRI Control Center Application Programming Interface (CCAPI) project and later standardized by IEC TC57 WG13 as part of the IEC61970 series standards for control centers. The CIM standard includes information associated with control center applications such as:

- Energy Management Systems (EMS)
 - Topology Processing
 - State Estimator
 - Power Flow
 - Security Analysis
- Supervisory Control and Data Access Systems (SCADA)
- Network planning

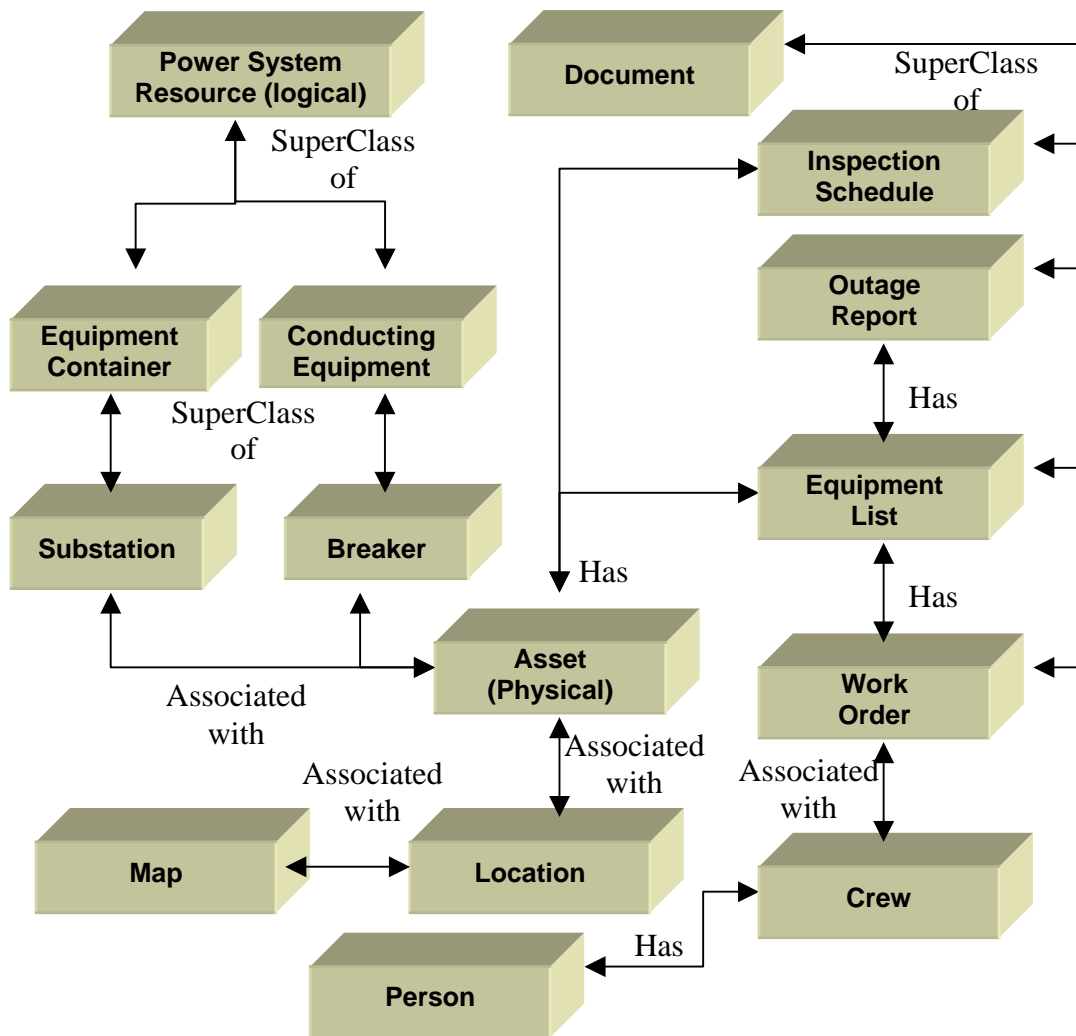
IEC TC57 WG14 has extended the CIM in their IEC61968 standard for Distribution Management Systems (DMS) related functions. IEC61968 added information models associated with operational support applications such as:

- Asset Management Systems (AMS)
- Work Management Systems (WMS)
- Construction Management
- Distribution Network Management
- Geographic Information Systems (GIS)
- Outage Management

The CIM describes real world objects in terms of classes, attributes and relationships. For example, the diagram below depicts the relationship between a set of CIM classes per IEC61970. A substation can contain voltage levels. Voltage levels can contain equipment. Breakers and Transformers are subtypes of a more general class called Conducting Equipment. Breakers have terminals that are associated with measurements. Transformers have windings that are also associated with measurements. And so on.



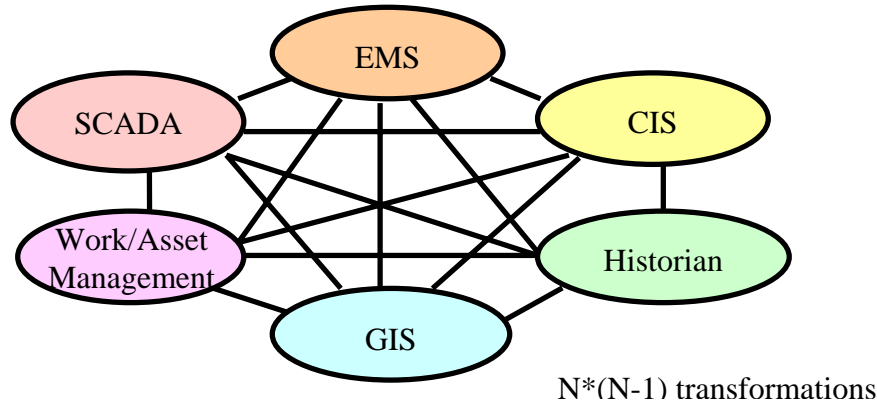
This diagram above does not illustrate all the possible associations specified in CIM. For example, substations, breakers, and transformers may also be directly associated with measurements. The diagram below illustrates a very simplified view of some of the CIM classes added by IEC61968:



In the diagram above, Power System Resource is the parent class of all logical equipment, such as circuit breakers, and equipment containers, such as a substation. In the CIM, the term “asset” refers to a physical object. Assets are associated one to one with logical equipment. Assets exist at a location that can be represented on a map. Elsewhere, the IEC61968 CIM also defines a parent document class. Outage reports, equipment lists, work orders, and inspection schedules are sub types of the document class. An outage report contains an equipment list that refers to one or more assets. And so on.

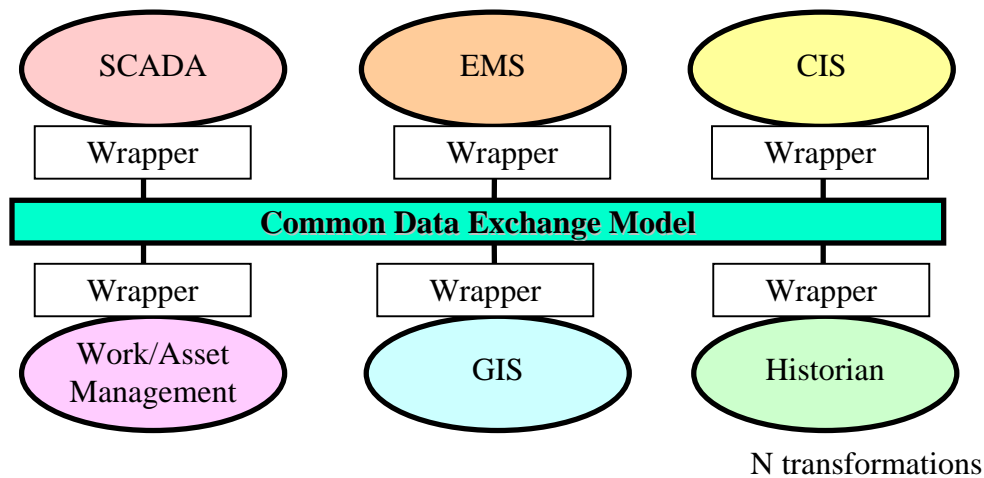
It is important to note that the CIM does not model utility data in an application-specific way. The CIM is used to model data aggregated by many different applications and not what is modeled by a single application internally. Without a common model by which to exchange data,

utilities are often required to perform many custom data transformations in order to integrate applications, as illustrated below:



Point-to-point Model Transformations

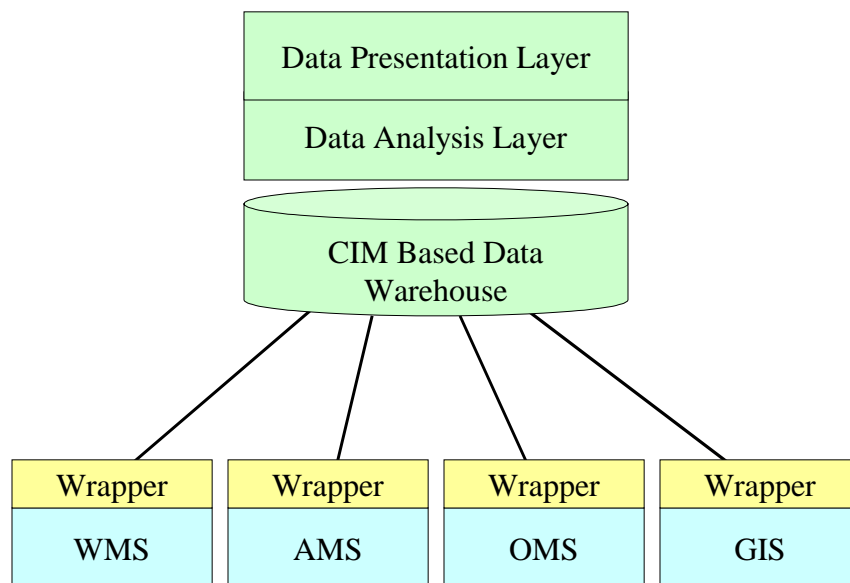
With a common model, each application communicates using the same language. This reduces the number of data transformations required from $N * (N-1)$ to N . Integrating legacy applications typically involves the creation of application wrappers that map legacy data formats to a common one.



Use of a common exchange model

The diagram above shows how operational needs (that is tactical as opposed to strategic goals) can be met using the CIM. In this case, real-time cooperation of applications allows business processes to be automated. However, the object classes represented in the CIM are abstract in nature and may be used in a wide variety of scenarios. The use of the CIM goes far beyond its use for application integration. The issues associated with construction of a data warehouse are very similar to application integration. That is, one needs to unify information from a variety of

sources so that analysis applications can produce meaningful and repeatable results. This standard should be understood as a tool to enable integration in any domain where a common power system model is needed to facilitate interoperability of data. The diagram below illustrates how the CIM can be used as the model for a data warehouse.



The diagram above illustrates a data warehouse that aggregates data from a collection of applications. The data is then analyzed and presented to the user for the purpose of making strategic decisions. The CIM provides an ideal starting point when designing the schema of the data warehouse because analysis requires a single comprehensive and self-consistent model. In this case, the wrappers extract, transform, and load application data into the data warehouse. It should be noted however, that almost all wrappers developed to date as part of a warehouse project are developed independently from the wrappers used for application integration. The reason this inefficiency was allowed to occur is that application integration projects have historically involved different stakeholders and software vendors. Furthermore, no vendor independent standards have existed for wrapper development. However, now at last, standards have been developed designed to allow a single wrapper to be used for both purposes.

As described above, plug and play also requires a common technical mechanism by which applications connect and expose information. In fact, it is agreement on common technical mechanisms that fully enables the creation of a single set of wrappers for application integration as well as data warehousing.

Common Exchange Mechanisms

The mechanism used to exchange data is determined by an application's interface. However, the native interface provided by an application is typically limited. For example, often legacy interfaces do not provide a means to discover what data is processed by a particular component at run time other than a rudimentary listing of legacy IDs (legacy IDs are typically referred to as

“tags”). Furthermore, legacy data cannot typically be viewed within the context of an inter-application data model such as a view of a power system network model. Legacy application interfaces most often are accessed using a variety of interface technologies including:

- RPC/API based (CORBA, COM, Java, C language)
- File based
- W3C Web Services/XML/HTTP based
- RDBMS/SQL based

Typically legacy interfaces:

- Do not expose data within the context of a common inter-application data model.
- Do not provide a means to discover what business object instances are serviced by a particular component instance other than a rudimentary listing of legacy IDs (tags) that cannot be viewed within the context of an inter-application data model such as a power system network model.

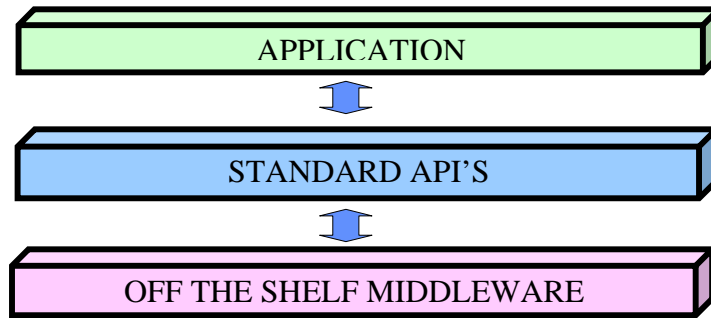
Without a means to discover what data an application processes, plug and play is nearly impossible to achieve. To address these impediments to plug and play and the need for a common exchange mechanism, or “how” data is exchanged, WG13 is in the process of adopting a series of interface standards called the Generic Interface Definition (GID). The GID is an umbrella term for four interfaces:

- **Generic Data Access (GDA)** – A generic request/reply oriented interface that supports browsing and querying randomly associated structured data – including schema (class) and instance information.
- **Generic Eventing and Subscription (GES)** – A publish/subscribe oriented interface that supports hierarchical browsing of schema and instance information. The GES is typically used as an API for publishing/subscribing to XML formatted messages.
- **High Speed Data Access (HSDA)** – A request/reply and publish/subscribe oriented interface that supports hierarchical browsing and querying of schema (class) and instance information about high-speed data.
- **Time Series Data Access (TSDA)** – A request/reply and publish/subscribe oriented interface that supports hierarchical browsing and querying of schema (class) and instance information about time-series data.

The table below organizes the GID functionality into a simple matrix:

	Generic	High Speed	Time Series
Request/Reply	GDA	HSDA	TSDA
Publish/Subscribe	GES	HSDA	TSDA

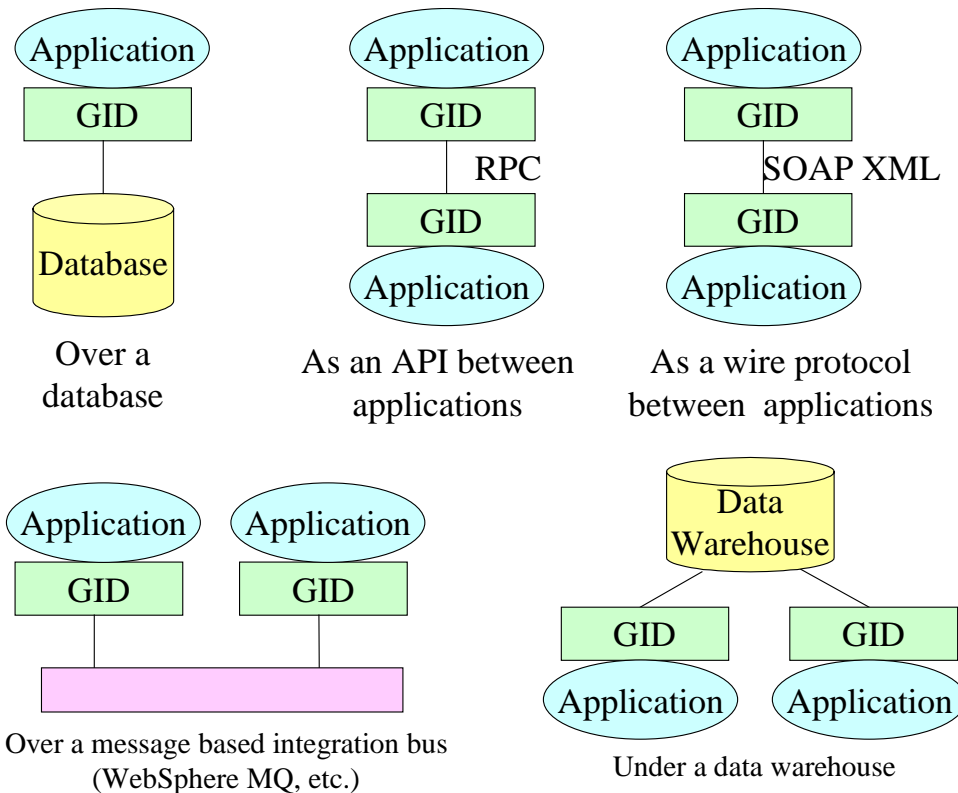
Applications use the standard interfaces to connect to each other directly or to an integration framework such as a message bus or data warehouse. The GID interfaces allow applications to be written independently of the capabilities of the underlying infrastructure.



The GID can be realized using a variety of middleware technologies including:

- RPC/API based CORBA, COM, Java, or C language specializations
- W3C Web Services/XML/HTTP based

Thus, the standard interface can be deployed as an API or as a wire level protocol such as Web Service based messaging. The diagram below includes several scenarios depicting how the GID might be used:



Regardless if these interfaces are implemented as an API or on the wire, the GID provides the following key functionality required for creation of a plug and play infrastructure:

- Interfaces are generic and are independent of any application category and integration technology. This facilitates reusability of applications supporting these interfaces.

- Interfaces support schema announcement/discovery – The schemas are discoverable so that component configuration can be done programmatically at run time. Programmatically exposing the schema of application data eliminates a great deal of manual configuration.
- Interfaces support business object namespace presentation – Each component describes the business object instances that it supports within the context of a common namespace shared among all applications such as a power system network model like the EPRI Common Information Model (CIM). It is not enough to merely expose the application data schema, one must also expose what specific breakers, transformers, etc, an application operates on. This also eliminates manual configuration as well as provides a means for a power system engineer to understand how enterprise data is organized and accessed.

The advantage of using generic interfaces instead of application-specific ones cannot be over emphasized. The benefits of using generic interfaces include:

- The interfaces developed are middleware neutral and have been designed to be implemented over commercially available message bus and database technology. This means a single wrapper can be used regardless on the technology used to perform integration.
- As application category independent, the same interfaces are used to wrap any application. This means that new wrappers do not need to be developed every time an application is added to the system.
- Creates a consistent and easy to use integration framework by providing a unified programming model for application integration.
- Enhances interoperability by “going the last mile”. Agreement on the “what” of data is not enough to ensure component interoperability. We also need to standardize on “how” data is accessed. To provide a simple analogy, we standardize on a 110/220 volt 60 hertz sine wave for residential electrical systems in the US. This is a standardization of “what”. However, we also standardize the design of the plugs and receptacles. This is a standardization of the “how”. The standardization of plugs and receptacles means that we don’t need to call an electrician every time we want to install a toaster. Similarly with software, standardizing on the interface means a connector does not need to be created from scratch every time we install a new application.
- Since application vendors can “shrink wrap” a CIM/GID compliant wrapper, the use of the CIM and GID can lower the cost of integration to utilities by fostering the market for off-the-shelf connectors supplied by application vendors or 3rd parties. The time and money associated with data warehousing/application integration wrapper development and maintenance is high. Typically, most money spent on integration is spent on the wrappers. An off-the-shelf CIM/GID wrapper can replace the custom-built “Extraction and Transformation” steps of an Extraction/Transformation/Load warehouse process. The availability of off-the-shelf CIM/GID compliant wrappers is a key to lowering application integration and data warehouse deployment and maintenance costs very significantly.

The GID interfaces support viewing of legacy application data within the context of a shared model such as the CIM. The GID interfaces take full advantage of the fact that the CIM is more than just a collection of related attributes – it is a unified data model. Viewing data in a CIM context helps eliminate manual configuration and provides a means for a power system engineer to understand how enterprise data is organized and accessed. The GID interfaces allow legacy data to be exposed within a power system oriented context. This makes data more understandable and “empowers the desktop” by enabling power system engineers to accomplish many common configuration tasks instead of having to rely on IT personnel.

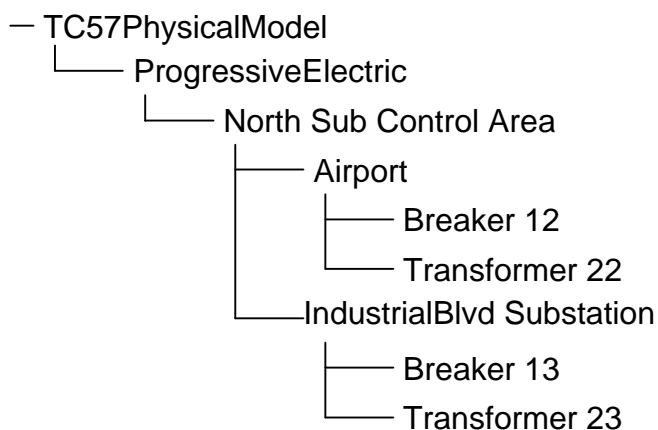
Namespaces

The GID interfaces specify two related mechanisms. The first specifies a programmatic interface that a component or component wrapper must implement. The second specifies how an information model, such as the CIM, is exposed via the programmatic interface. The later concept is embodied in the term “namespace”. Standard namespaces provide an agreement on how to communicate CIM based hierarchies via the GID and supply a utility specific way (CIM based) of viewing and configuring the exchange of data. That is, namespaces provide a means for exposing what schema and instance data an application processes.

Three namespaces are defined. They are:

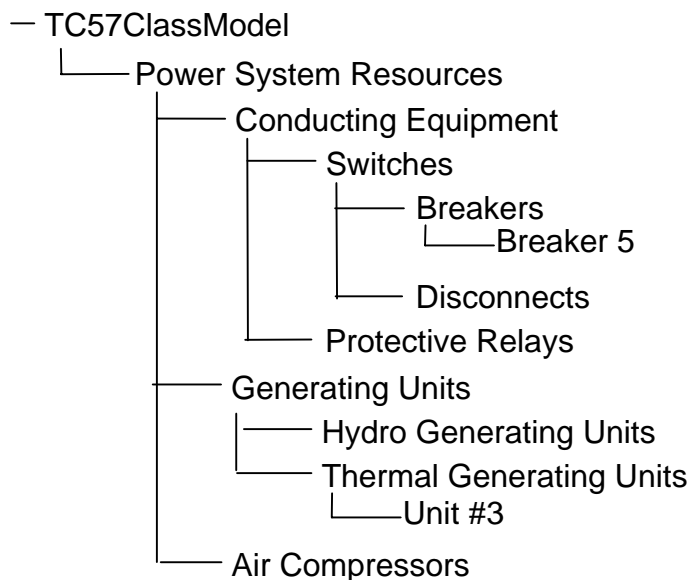
- TC57PhysicalModel
- TC57ClassModel
- TC57ISModel

The TC57PhysicalModel is a tree that orders power system related instance data in accordance to how it is contained from a physical perspective. Companies contain sub control areas, sub control areas contain substations, etc. The idea is that a power system engineer can find a breaker without having to remember a potentially convoluted or inconsistent naming scheme as shown below.



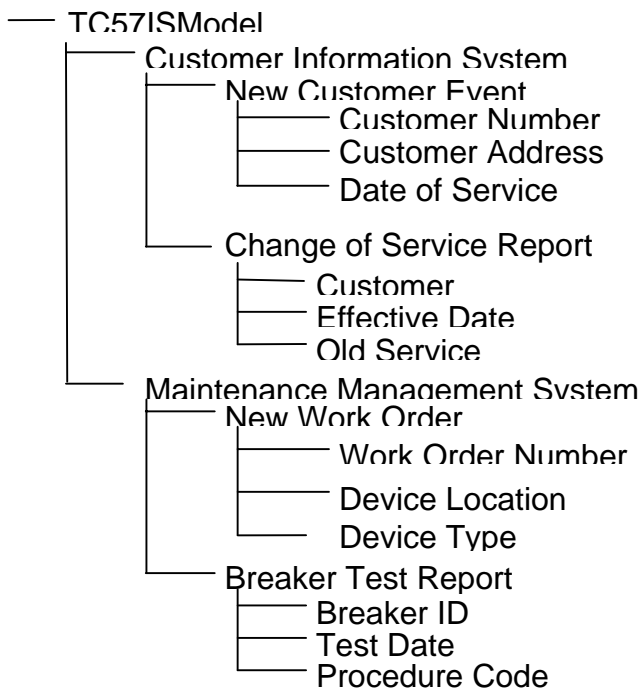
Example TC57PhysicalNamespace

The TC57ClassModel consists of a tree that orders power system related instance data in accordance to object types. Viewing data in this way is often most convenient for example when one wants to access all protective relays for example.



Example TC57ClassNamespace

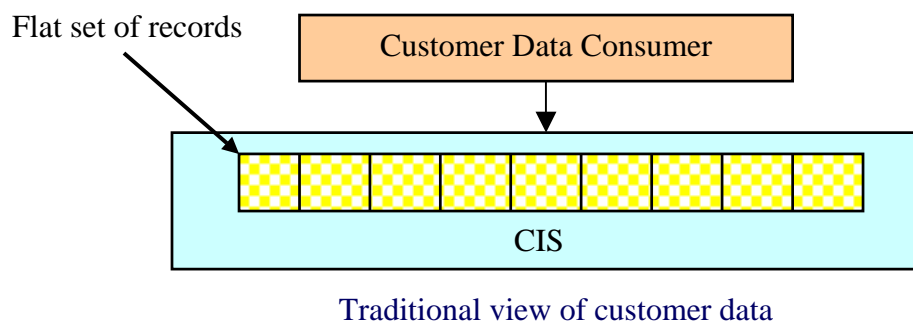
The TC57ISModel namespace is associated with the Generic Eventing and Subscription (GES) interface, the standard interface for publishing and subscribing described below. The tree allows an application to describe what message types (application data schema) it publishes as well as the content of each message type.



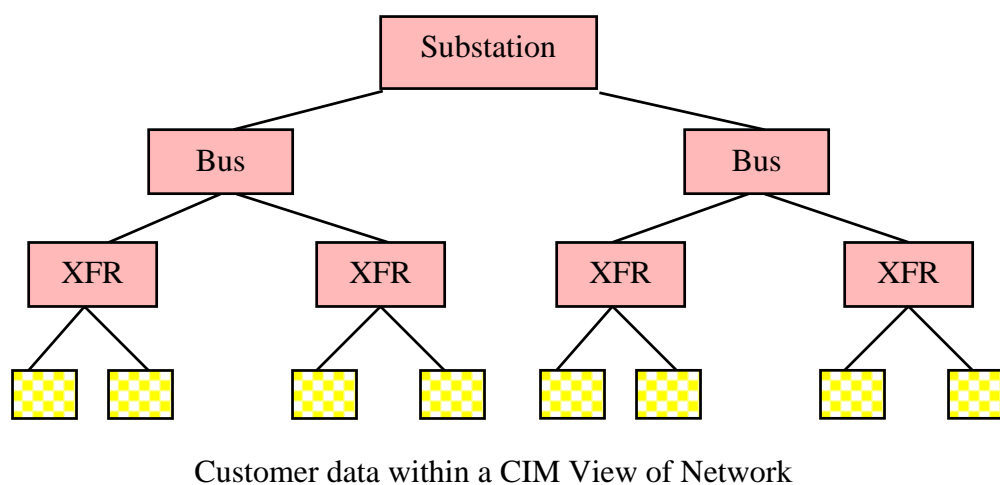
Example TC57ISNamespace

The use of namespaces

This section describes how a namespaces can be used at an application interface to make utility data more understandable. For example consider the diagram below:



This diagram shows how data is often presented by a typical legacy application. In this case customer data is presented by an application as a flat set of records without much information about how customers relate to information modeled in the CIM such as network topology. However, if one is interested in correlating the quality of power delivered to customer to a network element such as a transmission line for example, then it is useful to be able to put customer records in the context of the network model. The generic interfaces provide a standard way of exposing data such as customer records within a namespace, in this case the TC57PhysicalModel as illustrated below:

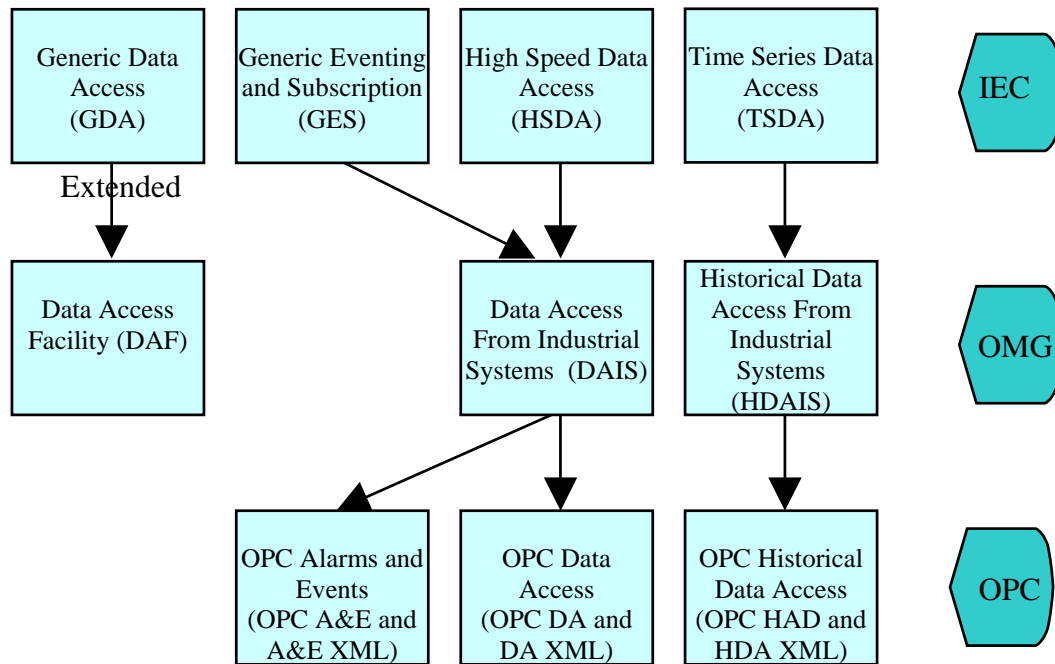


An additional benefit of providing customer data in a standard namespace is that off the shelf components can be created independently of individual customers requirements. Such

standardization is necessary if one hopes to foster a market for off the shelf wrappers for common applications.

The Generic Interfaces

Rather than inventing new interfaces, the IEC chose to leverage OPC - a set of widely deployed de facto standard interfaces frequently used in the process control industry and supported by close to 1000 vendors. Three out of four of the IEC interfaces incorporate the equivalent the OPC interface. However, since the OPC interfaces were originally based on Microsoft specific technology, the IEC also decided to leverage cross platform versions of the OPC interfaces standardized in the Object Management Group (OMG). The OMG is a software standards consortium consisting of all major software vendors. The OPC versions of the IEC interfaces are used when deploying COM, .Net or Web Services based interface technology so that the hundreds of OPC clients available off the shelf from many vendors are GID compliant today. On the other hand, the OMG version of the GID is use to define the CORBA, Java, or C Language interfaces¹. The diagram below illustrates the lineage between the WG 13, OMG, and OPC interfaces.

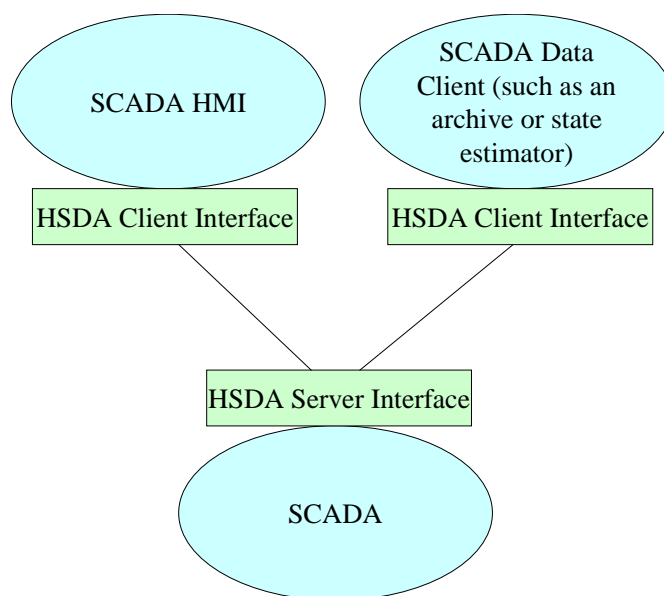


¹ For more information see IEC 61970 Parts 401 – 405.

While for the sake of convenience, this document will use the IEC names when discussing the Microsoft and non-Microsoft varieties of the interfaces, it should be noted that both the OMG and OPC variants of the interfaces are GID compliant².

High Speed Data Access

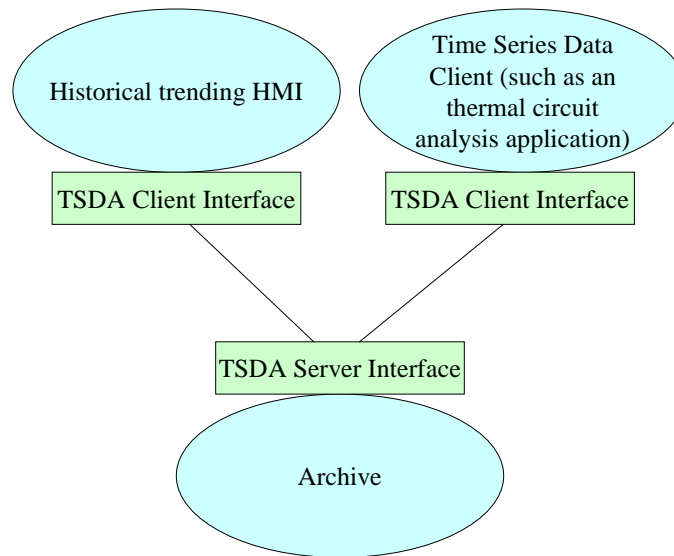
The High Speed Data Access (HSDA) interface was designed to handle the unique requirements of exchanging high throughput data. For example, a larger SCADA system might need exchange measurements at a rate exceeding 5,000 points per second. For these high performance situations, it is necessary to deploy an interface optimized for throughput. The tradeoff for achieving higher throughput is that HSDA is a small amount more time consuming to configure. HSDA supports the TC57 namespaces so that power system engineers can access measurement data in a user friendly way. The diagram below illustrates how HSDA might be deployed.



Time Series Data Access

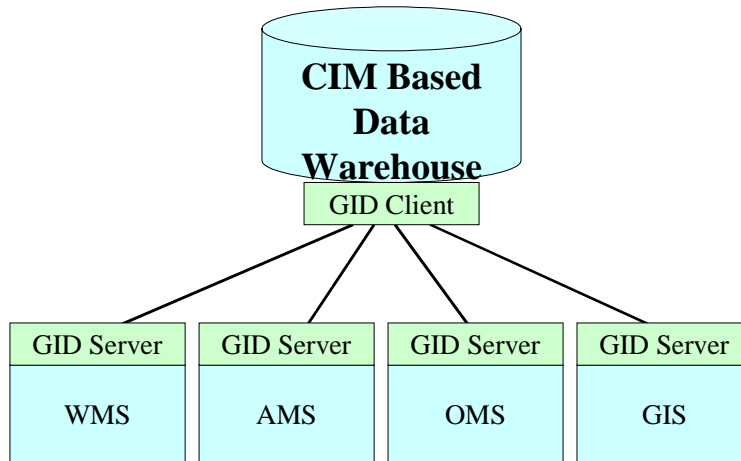
The Time Series Data Access (TSDA) interface is designed for the exchange of arrays of data where each array contains the values of a single data point over time. The mechanics of efficiently passing these arrays requires a separate interface. TSDA supports the TC57 namespaces so that power system engineers can access time series data in a user friendly way.

² All DAIS as well as OPC DA, HDA, and A&E clients are by definition compliant. All DAIS as well as OPC DA, HDA, and A&E servers are compliant if they present their data within the context of the TC57 Namespaces.



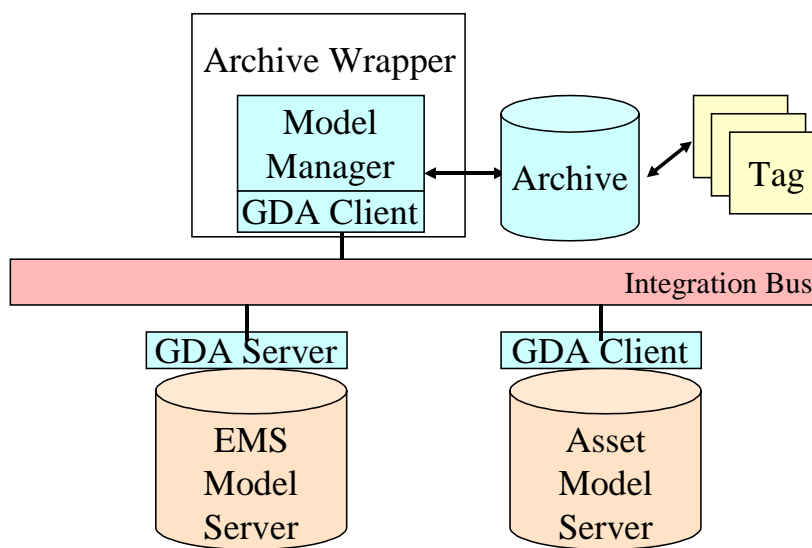
Generic Data Access

The Generic Data Access (GDA) interface specification is the one standard interface that was not created from a previously existing OPC specification. GDA provides Read/Write access to data typically in a database. It is similar in functionality to ODBC but is platform and schema neutral. The GDA more effectively leverages the work of NERC's Security Coordinator's Model Exchange Format (CIM XML). Using the GDA, a user accesses the data via the CIM terminology of classes, attributes, and relationships. Unlike ODBC, the GDA interface is independent of how data may be physically stored in the database. As a result, it is an ideal way for vendors to expose their data in a CIM compliant way that is database schema neutral so to enable the construction of a data warehouses as shown below:



GDA includes the ability to notify clients when data has been updated in the server. This functionality provides an important piece of the puzzle when constructing an infrastructure that enables a single point of update for model changes. For example, changes in an EMS modeling server can be used to drive the configuration of an archive or implement a synchronization routine with an asset management system as shown below.

Configuration Synchronization: Adding a Breaker Use Case



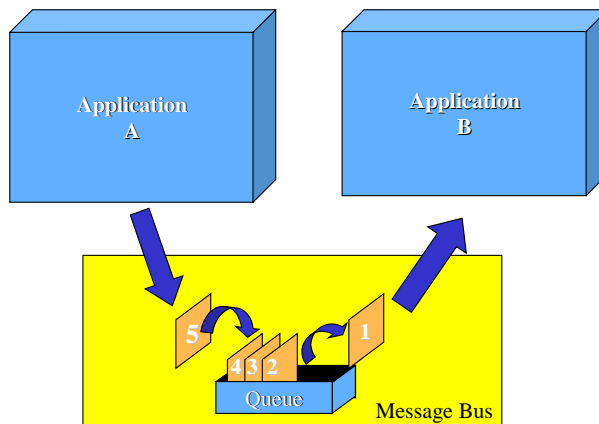
1. A breaker is added to the EMS using the model server.
2. User is prompted to map new logical breaker to new or existing physical breaker in asset model.
3. The archive wrapper also is notified.
4. Archive wrapper determines appropriate configuration changes and adds this to the archive configuration.
5. CIM model subset in archive is automatically updated and archiving begins³.

Generic Eventing and Subscription

Recently, the software industry has realized that application integration can be facilitated via the exchange of eXtensible Markup Language (XML) messages. Just as HyperText Markup Language (HTML) has become the universal language of the Web, businesses have sought a similar language for describing business data. XML has been adopted by the World-Wide Web Consortium (W3C) and is rapidly becoming the preferred format for exchanging complex business data internally and between E-Commerce applications. Similar to HTML, XML allows the designer to create custom schema and describe how they are used and thus provides the facilities to create self describing messages. This capability is independent of transport mechanisms, calling conventions (the order in which parameters are passed or how data is returned), and data formats. This significantly reduces the size and complexity of legacy application wrappers. XML- formatted business data offers standard and extensible information formats, or packages, with which to exchange information internally and with other businesses.

But utilities still need a reliable mechanism to send and receive XML packages. To use a post office analogy, no one waits at the front door for the postman to arrive before mailing a package. Mailboxes provide a convenient method for storing letters until a mail truck comes along to pick up the mail and deposit the received mail. One could use email, but email has not been designed for efficient automation. Alternatively, message oriented middleware products help link applications. In general, these software products include a message broker. With message broker technology, a business application can send business messages to a broker message queue for later delivery. The messages are then picked up by the message broker and dispatched to other internal or external applications. Message brokers facilitate location and technology independence and have proven to be the best way to link loosely coupled legacy applications

³ It is important to note that in order for the archive to present historical data within a TC57 namespace, that only a relatively small portion of the EMS model be maintained in the archive wrapper model manager.

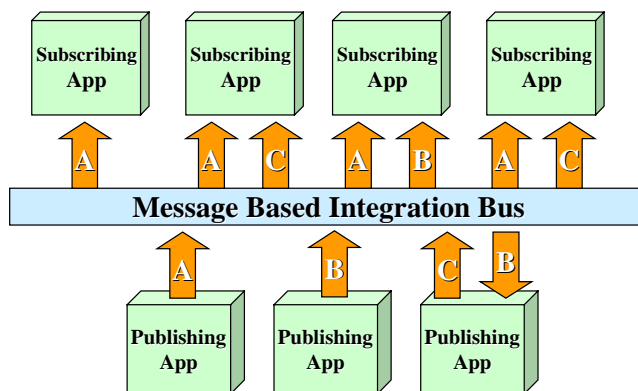


The use of messages to exchange data between applications

Persistent message queuing provides the basis for a robust application integration infrastructure because:

- Applications are decoupled in time from each other
- Provides fault recovery infrastructure

In addition to message queuing, a message based integration bus can enhance scalability via the use of publish and subscribe as shown below:



A, B, and C are topics

Publish/Subscribe

Using Publish/Subscribe, message sources post messages according to topics. Message consumers receive messages based on the topics they have subscribed to.

Publish/Subscribe decouples applications from data sources and facilitates scalability because.

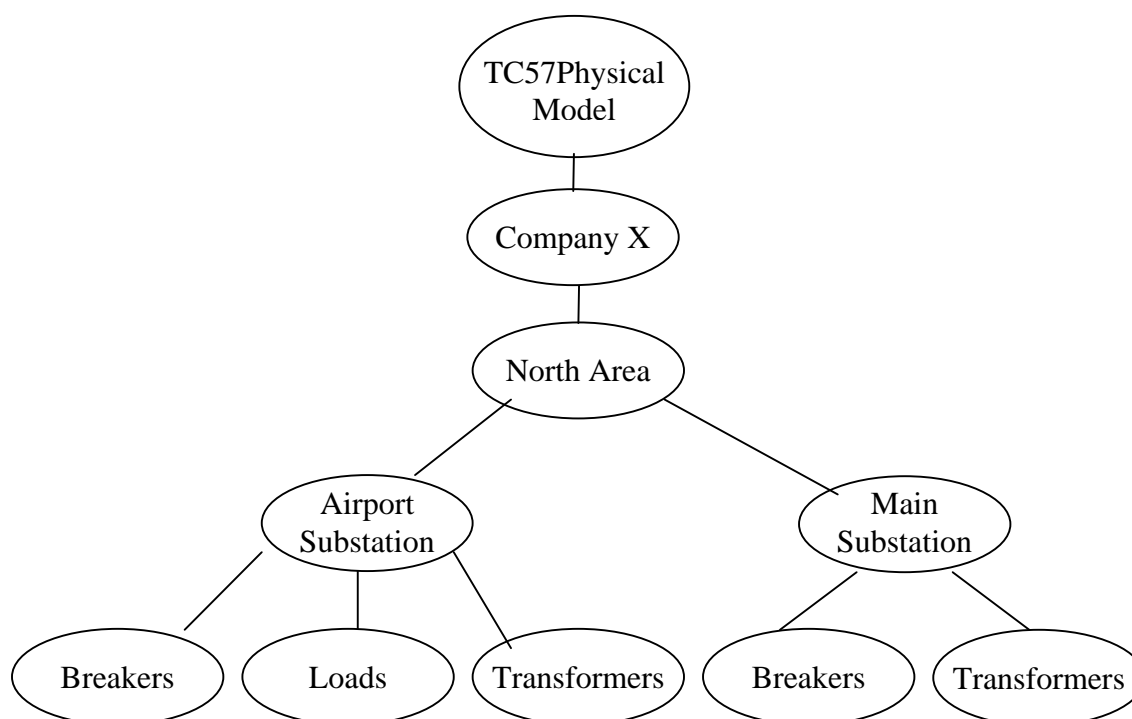
- Publishers do not need knowledge of subscribers
- Subscribers do not need knowledge of publishers
- Multiple subscribers can receive information without publisher configuration.

Publish/Subscribe supports redundancy and scalability:

- Multiple publishers can provide the same data
- More easily scalable for large systems as publishers only need to publish any given message once.

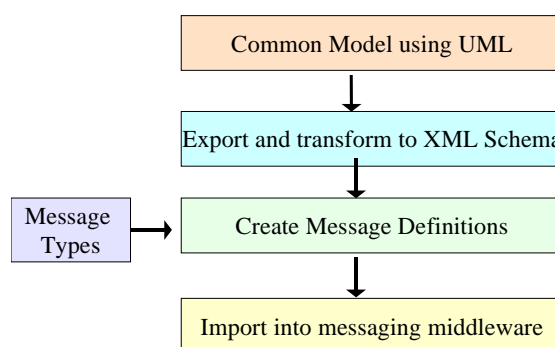
The Generic Eventing and Subscription (GES) interface specification is designed to be the primary mechanism for application integration. The GES provides an interface by which applications can publish and subscribe to CIM data. As a publish/subscribe oriented interface it supplies an ideal vendor-neutral interface for a generic application integration product such as IBM's WebSphere MQ or Tibco Rendezvous.

In addition to just providing a way for applications to publish or subscribe to data, the GES interface takes maximum advantage of CIM as presented in a TC57 Namespace. That is, GES provides a power system oriented mechanism for data subscription configuration that power system engineers can use. The diagram below illustrates a sample TC57PhysicalNamespace. The namespace would typically be displayed in a subscription configuration graphical user interface (GUI). By displaying a view of the power system model in a message subscription configuration GUI, the user can set up subscriptions without having to know the often complex subscription configuration script syntax used by the generic application integration tool. Off the shelf generic integration tools know nothing about power system models. By providing a power system specific user-friendly layer on top of the generic application integration tool, power system engineers can do things such as subscribe to a daily report without requiring the assistance of an information technology professional.



Using the example namespace above, the user could subscribe to data related to a company, to just a particular substation, or just a set of devices in a substation – potentially all done via a user friendly GUI.

Besides providing a context for subscribing to messages, the CIM provides a data dictionary for GES messages. In fact, WG 14 has defined a set of message definitions for common utility operational business processes. The figure below illustrates the process of defining messages from the CIM.



Model Driven Message Definition

Architectures for the standard interfaces

The GID has been designed to create what is called an Enterprise Service Bus (ESB). An ESB solves many problems associated with traditional use of a message bus as well as component based (COM/DCOM, CORBA, Java) integration infrastructures. For example, traditional message bus based architectures may be:

- Expensive to deploy and maintain – These systems are often completely proprietary forcing the user to employ expensive tools and single source application wrappers.
- Cannot optimize individual links to meet project specific requirements – These systems have monolithic architectures. Users are forced to use the message bus for everything or develop their own custom application wrappers if the message bus is unable to meet performance requirements for a particular link.

On the other hand, traditional client server oriented component-based integration based on CORBA or COM/DCOM architectures may be limited by:

- Excessive coupling between clients and servers is inflexible – applications connect directly to one another so that a change to the server requires reconfiguration to the client.
- These technologies require a common security domain context, function calling convention, binary data types, and way of locating and activating remote applications. Additionally, CORBA and DCOM typically require that server applications must be ready to service a request when the client wishes. Thus CORBA and DCOM are better suited to assembly of tightly coupled components.

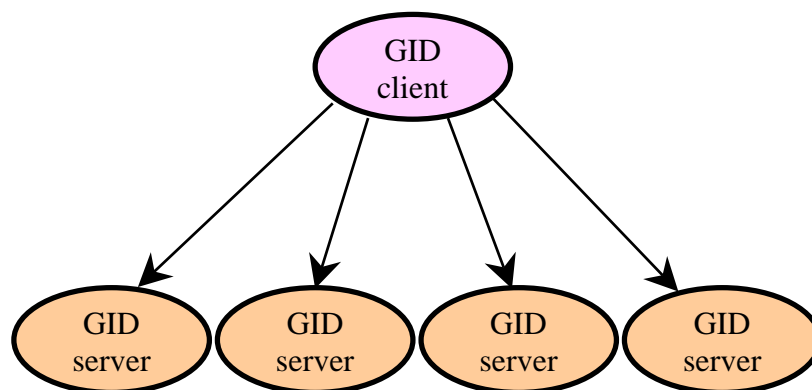
An ESB attempts to address the limitations of traditional message bus and component based infrastructures by relying on agreed upon non-proprietary interfaces to an message based integration bus. Often ESB's employ a Web Service based infrastructure. The advantages of this approach include:

- Low cost – Web Service infrastructure can be supplied by platform provider (Linux, IBM, Microsoft, Sun, etc.)
- Interoperability – multiple vendors including all major application integration tool vendors support Web Services.

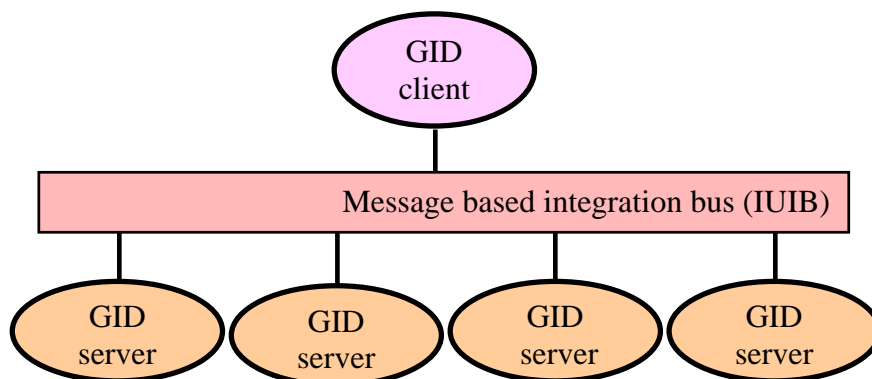
However, a traditional Web Service based infrastructure has some disadvantages:

- Standards not fully complete.
- Inability to easily aggregate multiple services – clients typically connect to every server instead of simply connecting onto the integration infrastructure because there is no single agreed upon set of generic interfaces.

A CIM/GID ESB however, addresses all of the above limitations by remaining transport neutral. In this way, Web Services, COM, CORBA, or some proprietary transport such as MQ Series may be used where appropriate. With regard to aggregating services, the difficulty stems from the fact that an ESB is client/server in nature. Therefore, if a client needs to communicate to multiple data sources, it must connect to each one individually as shown below.



However, experience has shown that an integration architecture based on point-to-point links such as these is inflexible and difficult to maintain. Instead, the GID based infrastructure typically relies on a proxy-based architecture enabled by the generic nature of the GID services. The bus acts as a proxy by joining the namespace of all servers. From the clients point of view there is only one server of any particular type (GDA, GES, HSDA, TSDA). Thus point-to-point links are eliminated.



The existence of an aggregating services has important ramifications with regard to application integration including:

- Location independence between applications means that applications can be tied together at run time and not at compile time - greatly reducing maintenance.
- Application integration or data warehousing infrastructure can manage the common schema and provide a run time environment for use of distributed objects - application development becomes simplified.

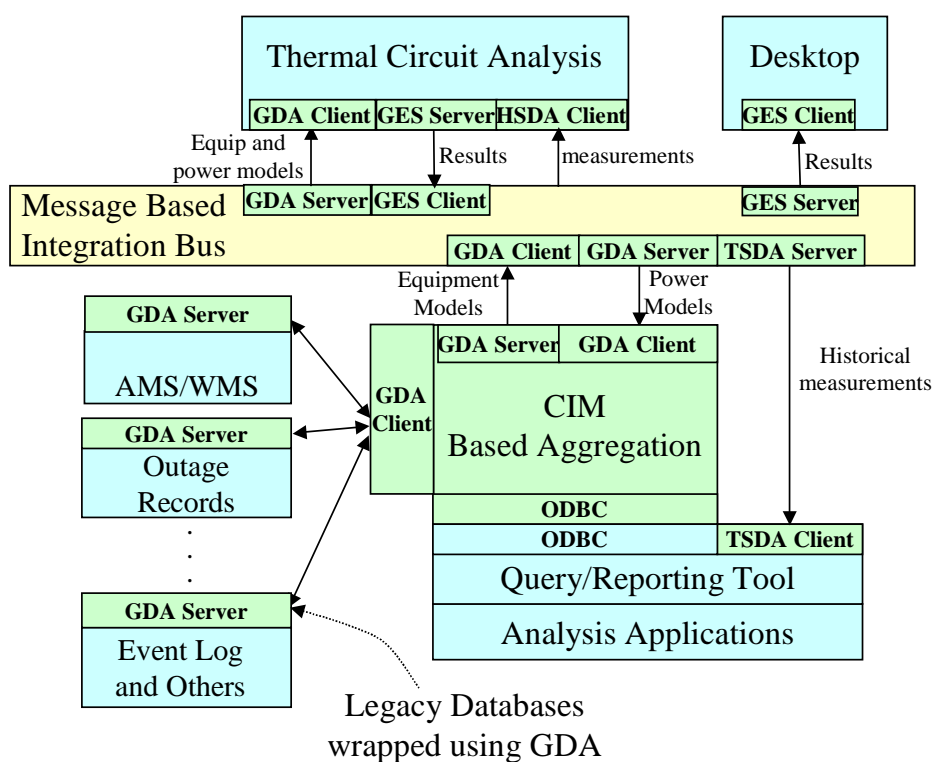
Deploying the standard interfaces

This section describes how a deployment of the CIM and GID can be used to create a platform for data warehousing and application integration. In this case, we consider two projects with apparently different goals. The projects are:

- Substation asset data analysis. This development integrates the following data:
 - Asset/Equipment data
 - Historical measurements
 - Power system network models
- Control center application integration. This development integrates the following applications:
 - Legacy EMS

- New archive, state estimator, and power flow
- CIM based power system modeling environment

By sharing a common design framework, a message based application integration and data warehousing solution can be built simultaneously. This approach reuses shared application wrappers to leverage the investment in each without requiring all data to be copied to a data warehouse. Separately, the cost of developing individual wrappers for data warehousing and for application integration can be prohibitive. By exposing application wrappers directly to the data analysis infrastructure without requiring an intervening copy of all the data in a intermediate warehouse, flexibility is maximized while costs are minimized. The diagram below illustrates the asset analysis project components.



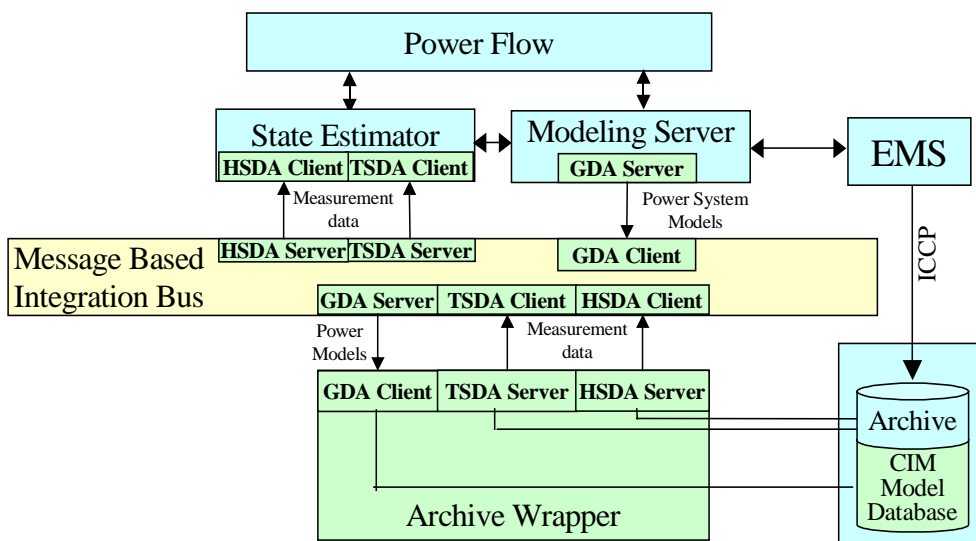
Asset Analysis Project

In this diagram, a collection of databases including an AMS/WMS, transformer outage information, transmission breaker change events (Event Log), and others are integrated using GDA. The databases are aggregated with power system modeling data supplied by the message based integration bus. The databases are tied directly to the Query/Reporting Tool and not to the message bus for performance. By avoiding the XML messaging required by the message bus and only using the binary interface-to-interface remote procedure calls, data warehouse query performance is maximized. This architecture highlights one of the advantages of using a transport neutral interface such as GDA. In this architecture, links are optimized to meet project goals while still enabling a single standard off-the-shelf wrapper for applications. Thus,

application vendors can supply a single standard wrapper for data warehousing and message based application integration.

Besides the warehouse, analysis applications in this use case include deployment of a thermal circuit analysis program. Rather than run on top of a vendor specific analysis platform, the analysis application connects directly into the CIM/GID integration infrastructure using the GID interfaces. Periodically, the thermal analysis examines current loading and temperatures and after running calculations, publishes results on to the bus. The thermal analysis obtains required asset and power system information about equipment from the integration bus GDA server.

The control center project involves the integration of transmission related applications using the integration bus as shown below:

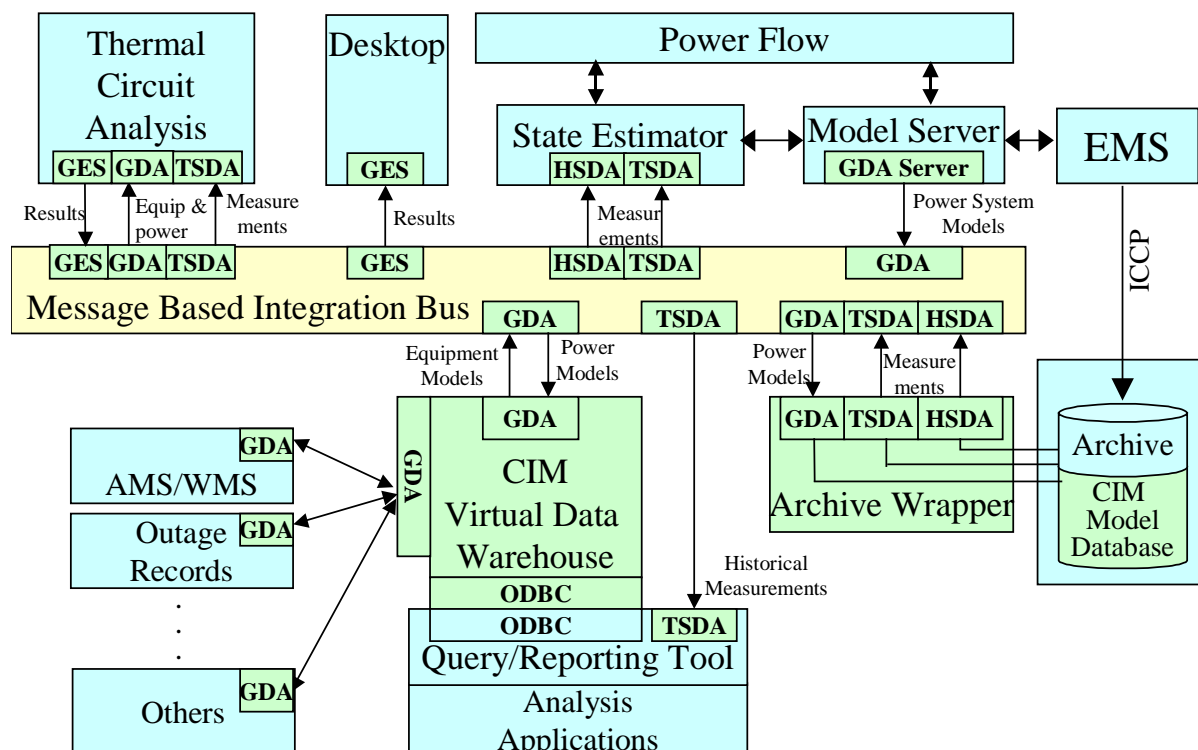


Control Center Project

As mentioned above, the principle motivation for the control center project is to facilitate integration of control center applications. In this case, a new state estimator, a power flow and an archive application are being integrated with the legacy EMS using a CIM based modeling tool. The modeling tool supplies power system modeling information to the state estimator and power flow application via proprietary mechanisms. The modeling tool also supplies power system model data to the asset analysis applications as well as some of power system model data to the archive wrapper via a CIM/GID interface.

As discussed previously, an application uses the GID to expose information within the context of the CIM. In this case, the archive wrapper imports a small amount of the power model so that it can expose archive data within a CIM context.

The diagram below depicts the combined system.



Combined Architecture

Benefits

This section summarizes the benefits of the approach described in this report.

Standards

It is said that the average age of utility employees in the US is close to 50 years old. If one combines the number of people that will retire over the next 5 to 10 years with average utility power system engineer turnover, then it is a fairly safe assumption that utilities need to carefully consider how system knowledge continuity will be accomplished. This situation is compounded by the fact that many utilities rely on systems that are customized for their particular installation. For example, many utilities model a network differently even though they may use the same modeling tool as their neighbor. A solution to this problem can include agreement by utilities on

standardized best practices. Not only will agreement between utilities enlarge the knowledge pool so that more effective operations can be accomplished, it will allow utilities to wean themselves off customized solutions. The CIM and GID can play an important role in the move towards non-biased standardized solutions. The ESB architecture presented here is entirely based on standardized interfaces – free of political tugs-of-war and vendor lock-in. Adoption of this architecture can help provide the continuity that utilities need.

3rd party applications

Standardization of the CIM and GID fosters interoperability of components for many uses. One market that will likely be created as a result of this standardization is the market for CIM based asset analysis applications or EMS add-ons. Today, every analysis applications or add-on must be extensively customized for every deployment. If an analysis or add-on application supplier can assume the existence of the CIM and GID, then the supplier can sell the same tool to different utilities with a minimal amount of customization. Decreased development cost, together with competition, should help drive down prices.

Off the shelf wrappers

As stated above, the cost of application wrappers for data warehousing and application integration is the most significant cost of deploying these technologies. An integration infrastructure based on the CIM/GID standards described here helps enable the availability of off-the-shelf wrappers because application vendors and 3rd parties can now reasonably expect that a CIM/GID solution developed for one location could be used in others too. Furthermore these wrappers can be deployed independently of what integration infrastructure the utility happens to choose.

Incremental approach

Fundamentally, application integration with the CIM and GID involves looking at the big picture. However, integration may encompass data from a large or small set of applications. One does not need to undertake a major project that requires many months to complete. The issue here is the development of a long-term enterprise wide integration strategy so that a small integration project does not become just another slightly larger island of automation. Thinking at the enterprise level while integrating at the department level minimizes risk and maximizes the chances for long-term success. Within the context of a long-term plan, the CIM and GID makes it possible to tackle the problem of integration in a staged approach. Rather than having to understand the relative mapping of all of the CIM for each application that you may need in the future, the CIM and GID lets you start with a more focused approach and expand the solution over time. For example, the GID can be used to provide a CIM wrapper on existing data warehouses. You could start with one existing data warehouse to enable the CIM based integration of only that specific data warehouse. You can then increase the scope of the CIM and GID incrementally until, eventually, all data in the various data marts, warehouses and applications are all available via a unified CIM view. The inevitable inconsistencies in meaning or content between existing databases and applications can be gradually discovered and addressed as they need to be. In this manner, the CIM and GID delivers incremental value with staged effort throughout the process.

Towards Plug and Play

The sections above have described how the use of the CIM and the GID can allow components to connect and exchange information automatically. However, there is one problem that cannot be resolved via standards. The remaining problem is that every utility typically names objects such as a breaker, substation, or any other resource in a non-uniform manner. The ID used to refer to an individual breaker in one application will frequently not match the name in a second application. To get the two applications to exchange data, a name mapping must be created. Fortunately, creating a name mapping table does not require custom programming and can frequently be partially automated by using the name conventions that do exist within most applications.

Conclusion

The benefits of standardized data models and component interfaces are clear. Utilities can significantly lower the cost of performing integration by leveraging off-the-shelf components and wrappers from application vendors or third parties. Furthermore, the CIM and the standard interfaces provide a power system specific mechanism to more easily deploy, configure and use an integration bus or data warehouse. As a result, a utility can achieve greater efficiencies and adaptability.

References:

The following standards documents are available at the time of writing.

For the latest publicly available WG 13 documents see:
ftp.kemaconsulting.com/epriapi/downloads. They include:

- 61970 Part 1: Guidelines and General Requirements
- 61970 Part 2: Glossary
- 61970 Part 301 Common Information Model
- 61970 Part 402 Common Services
- 61970 Part 403 Generic Data Access
- 61970 Part 404 High Speed Data Access
- 61970 Part 405 Generic Eventing and Subscription
- 61970 Part 407 Time Series Data Access
- 61970 Part 501 CIM RDF Schema
- 61970 Part 503 CIM XML Model Exchange Format

For the latest publicly available OMG documents see: www.omg.org. They include:

- OMG, Utility Management System Data Access Facility, document formal/2001-06-01
- OMG, Data Access from Industrial Systems Specification, Draft Adopted Specification, OMG document: dtc/01-07-03, July, 2001
- OMG, Historical Data Access from Industrial Systems, Request for Proposal, OMG document, January 31, 2002.

For the latest publicly available OPC documents see: www.opcfoundation.org They include:

- OPC Data Access Custom Interface Specification, Version 2.05, OPC file: opcda205_cust.doc, December 17, 2000
- OPC Alarms and Events Custom Interface Specification, Version 1.02, OPC file: opcae102_cust.doc, November 2, 1999
- OPC Historical Data Access Custom Interface Standard, Version 1.1, OPC file: opc_hist_cust.doc, January 26, 2001

For the latest publicly available WG 14 documents see: <http://www.cimuser.com> They include:

61968 Part 1: Interface Architecture and General Requirement

61968 Part 2: Glossary

61968 Part 3: Interface Standards for Network Operation

61968 Part 4: Interface Standards for Records and Asset Management

61968 Part 5: Interface Standards for Operational Planning and Optimization

61968 Part 6: Interface Standards for Maintenance and Construction

61968 Part 7: Interface Standards for Network Extension Planning

61968 Part 8: Interface Standards for Customer Inquiry

61968 Part 9: Interface Standards for Meter Reading and Control

61968 Part 10: Interface Standard for Systems External to, But Supportive of, Distribution Management

61968 Part 11: Common Information Model

Appendix

As discussed above, WG 14 has defined a set of messages for business process automation. However, WG 14 does not define any particular infrastructure but instead defines a set of verb (functions) and noun (message) pairs. Central to how WG 14 defines business processes is a special application called a master system. A master system is one that is responsible for creating, deleting, as well as maintaining state for a particular type of resource. For example, a purchasing application is responsible for creating and deleting purchase orders and an EMS modeling tools is responsible for maintenance of logical device related resources associated with a transmission system. The compliment of a master system is a driven system. For example, an archive acts as a driven system because much of its configuration can be driven by the EMS. Sometimes, an application will act as both a master and a driven system. For example, an AMS is responsible for maintaining asset records. However, an AMS may act as a driven system with respect to an EMS if the AMS maintains a mapping from physical to logical devices.

WG 14 defines a set of verbs to cover data exchange model from a master system point of view and from a driven system point of view. Specifically, one set of present tense verbs used by a driven system purpose request action by the master system and another set of past tense verbs used by a master systems for the publishing when a resource has been created, deleted, or changed. Present tense verbs used to act on master systems (the system of records for the given resource) will result in a document or resource to be created or updated in the master system of that document if the request is processed successfully by the master system. Past tense verbs used to act on the driven systems will result in all referenced and/or replicated documents to be updated. .

Verbs are short hand way of referring to GID services in process flow docs. At the time of actual component interface development and deployment, verbs can be replaced with the corresponding service. This section describes the mapping between WG 14 verbs and WG 13 services.

The present tense verb used to act on a master system include:

Verb	Meaning	GID Service Used
CREATE	The CREATE verb is used to request to the master system to create a new document. The master system may in turn publish the new document using the verb CREATED. The master system may also use the verb REPLY to response to the CREATE request, indicating whether the request has been processed successfully or not.	GID CDA Update: create_resource. Reply is handled via exception. That is a reply is only sent by the master in the event that a service invocation has failed.
CHANGE	The CHANGE verb is used to request to the master system to make a change in the document based on the information in the message. The master system may in turn publish the changed document using the verb CHANGED to notify that the document has been changed since last published. The master system may also use the verb REPLY to response to the	GID CDA Update: set_values. Reply is handled via exception. That is a reply is only sent by the master in the event that a service invocation has failed.

	CHANGE request, indicating whether the request has been processed successfully or not.	
CANCEL	The CANCEL verb is used to request to the master system to cancel the document. The master system may in turn publish the canceled message using the verb CANCELED to notify that the document has been canceled since last published. The master system may also use the verb REPLY to respond to the CANCEL request, indicating whether the request has been processed successfully or not. The CANCEL verb is used when the business content of the document is no longer valid due to error(s).	GID CDA Update: set_values. Reply is handled via exception. That is a reply is only sent by the master in the event that a service invocation has failed.
CLOSE	The CLOSE verb is used to request to the master system to close the document. The master system may in turn publish the closed message using the verb CLOSED to notify that the document has been closed since last published. The master system may also use the verb REPLY to response to the CLOSE request, indicating whether the request has been processed successfully or not. The CLOSE verb is used when the business document reaches the end of its life cycle due to successful completion of a business process.	GID CDA Update: set_values. Reply is handled via exception. That is a reply is only sent by the master in the event that a service invocation has failed.
DELETE	The DELETE verb is used to request to the master system to delete the document. The master system may in turn publish the closed message using the verb DELETED to notify that the document has been deleted since last published. The master system may also use the verb REPLY to response to the DELETE request, indicating whether the request has been processed successfully or not. The DELETE verb is used when the business document should no longer be kept in the integrated systems either due to error(s) or due to archiving needs.	GID CDA Update: delete_resource. Reply is handled via exception. That is a reply is only sent by the master in the event that a service invocation has failed.
GET	The GET verb is used to request to the master system to get the current data for a given document reference code or a set of documents. The master system may in turn publish the document using the SHOW verb, if the document is available, or use the verb REPLY to response to the GET request, indicating that the document is not available.	DAF or GID CDA Filtered Query:

The past tense verbs used to act on driven systems include:

Verb	Meaning	GID Service Used
CREATED	<p>The CREATED verb is used to publish the creation of a document as a result of either an external request or an internal action within the master system of that document. This is the first time that data for this document reference code has been published as the result of internal or external request; in which case, it would use the same document reference as the CREATE message. This message type is usually subscribed by interested systems and could be used for mass updates. There is no need to reply to this message type.</p>	<p>If the message sent includes all of the data related to a created document, then this verb maps to the publication of an OPC/DAIS Simple Event. However, the DAF and GID CDA provide a mechanism where only a summary event is sent. Since the amount of data sent for a created event can be large, and create can occur frequently, publication of created messages can consume a great deal of the network bandwidth. Consequently, the DAF and GID CDA IDL provide this optimization.</p>
CHANGED	<p>The CHANGED verb is used to publish the change of a document as a result of either an external request or an internal action within the master system of that document. This could be a generic change in the content of the document or a specific status change such as “approved”, “issued” etc. This message type is usually subscribed by interested systems and could be used for mass updates. There is no need to reply to this message type.</p>	<p>If the message sent includes all of the data related to a changed document, then this verb maps to the publication of an OPC/DAIS Simple Event. However, the DAF and GID CDA provide a mechanism where only a summary of the changes is sent. Since the amount of data sent for a changed event can be large, and changes occur frequently, publication of change messages can consume a great deal of the network bandwidth. Consequently, the DAF and GID CDA IDL provide this optimization.</p>
CLOSED	<p>The CLOSED verb is used to publish the normal closure of a document as a result of either an external request or an internal action within the master system of that document. This message type is usually subscribed by interested systems and could be used for mass updates. There is no need to reply to this message type.</p>	<p>If the message sent includes all of the data related to a closed document, then this verb maps to the publication of an OPC/DAIS Simple Event. However, the DAF and GID CDA provide a mechanism where only a summary event is sent. Since the amount of data sent for a closed event can be large, and closes can occur frequently, publication of change messages can consume a great deal of the network bandwidth. Consequently, the DAF and GID</p>

		CDA IDL provide this optimization.
CANCELED	The CANCELED verb is used to publish the cancellation of a document as a result of either an external request or an internal action within the master system of that document. This message type is usually subscribed by interested systems and could be used for mass updates. There is no need to reply to this message type.	If the message sent includes all of the data related to a canceled document, then this verb maps to the publication of an OPC/DAIS Simple Event. However, the DAF and GID CDA provide a mechanism where only a summary event is sent. Since the amount of data sent for a cancel event can be large, and cancels can occur frequently, publication of cancel messages can consume a great deal of the network bandwidth. Consequently, the DAF and GID CDA IDL provide this optimization.
SHOW	The SHOW verb is used to publish the most current content of a document as a result of either an external GET request or an internal action within the master system of that document. This message type is usually subscribed by the requesting system(s) or other interested systems. There is no need to reply to this message type.	Publication of an OPC/DAIS Simple Event.
REPLY	The REPLY verb is used to publish the processing result of an external request to the master system to create, change, delete, cancel, or close a document. The REPLY message type could contain specific confirmation information as to whether the request is processed successfully or not and provide alternatives if applicable. This message type is usually subscribed by the requesting systems. There is no need to reply to this message type.	Reply maps to an exception thrown by a DAF or GID CDA Server

SUBSCRIBE	The SUBSCRIBE verb is used to publish the request to ask the master system of a document to publish a CHANGED document whenever there is a change to the document. It implies that	OPC/DAIS Subscription

	the master system will not publish the CHANGED document unless there are one or more subscribers for the changed information.	
UNSUBSCRIBE	The UNSUBSCRIBE verb is used to publish the request to ask the master system of a document to stop publishing a CHANGED document whenever there is a change to the document. It implies that the master system will not publish the CHANGED document only when there are no subscribers at all.	OPC/DAIS Subscription