INTRODUCING THE FIRST PART OF A NEW STANDARD FOR SYSTEM INTERFACES FOR DISTRIBUTION MANAGEMENT SYSTEMS (DMS)

E Lambert
Electricité de France
1, Avenue du General de Gaulle, BP408, 92141 Clamart Cedex (France)
Tel: +33 1 47654029 – Fax: +33 1 47653991– E-mail: Eric.Lambert@edfgdf.fr

W D Wilson
Schneider Ltd
PO Box 41, Chippenham, Wiltshire SN15 1JJ (UK)
Tel: +44 1249 456045 – Fax: +44 1249 659635 – E-mail: bill_wilson@schneider.co.uk

SUMMARY

This paper is presented on behalf of IEC TC57 (Power Systems Control and Associated Communications) Working Group (WG) 14 to provide an overview of how it is working to define internationally recognised standard System Interfaces for Distribution Management Systems (DMS). WG14's objective is to aid Utilities and Vendors by identifying and defining standard interfaces so that useful information can be electronically exchanged between different operational systems, thus reducing implementation and support costs. The standards are limited to the definition of the interfaces and are to be used across multiple computer platforms and languages. The authors, both members of WG14, gratefully acknowledge the contribution to this paper of the other members.

THE MISSION OF WG14

At the Plenary Meeting of the International Electrotechnical Commission Technical Committee 57: Power System Control and Associated Communications, held in Dresden in September 1996, the decision was taken to proceed with the work of drafting a new standard for System Interfaces for Distribution Management Systems. This followed the initial work-study that had already been undertaken by an Ad Hoc Working Group. Working Group 14 (System Interfaces for Distribution Management) is tasked to produce a series of standards named IEC 61968. The first part of this series was published at the end of 1998 in the form of a committee draft.

The IEC 61968 series is intended to facilitate inter-application integration, as opposed to intra-application integration, of the various distributed software application systems supporting the management of utility electrical distribution networks. *Intra-application* integration is aimed at programs in the same application system, usually communicating with each other using middleware that is embedded in their underlying runtime environment, and tends to be optimized for close, real-time, synchronous connections and interactive request/reply or conversation communication models. IEC 61968, by contrast, is intended to support *the inter-application* integration of a utility enterprise that needs to connect disparate applications that are already built or new (legacy or purchased applications), each supported by dissimilar runtime environments. Therefore, IEC 61968 is relevant to loosely coupled applications with more heterogeneity in languages, operating systems, protocols and management tools. IEC 61968 is intended to support applications that need to exchange data on an event driven basis. IEC 61968 is intended to be implemented with middleware services that will complement, not replace, utility data warehouses, database gateways, and operational stores.

Why a standard

Electricity distribution worldwide is entering a period of change. Two major aspects of the change are to be seen in:

- The electricity market enters 1999 in the deregulation process. At the same time consumers have open access to independent suppliers outside the distribution company's service territory.
- The increased awareness of business and residential consumers perception of the utility's operation, leading to a greater emphasis on quantifying the 'cost' of providing services and improvements on "perceived quality".

Electricity de-regulation, which is in various stages of implementation around the world, will force many electrical utilities to increase accuracy in managing their distribution networks and decrease their reaction time to events. To achieve this, the degree of automation of the distribution network will have to increase with a corresponding increase in the availability of tools to help the management of the process.

The utilities have to focus more on the needs and perceptions of their customers to achieve their business goals. Although the core business functions are unlikely to change dramatically they will need to be performed more efficiently and still meet consumer-oriented goals. The inevitable outcome is for the utilities to employ information technology to support improved efficiencies and right-sizing of their operations.

The current situation in utilities at the distribution level forces application developers to build their own interfaces to any systems from which they need information. This state of affairs makes building applications difficult because the developers have to be concerned not only with the implementation of the functionality but also with the implementation of the different interfaces they may need.

Different systems are involved in DMS (for instance, a trouble-call system, a geographical information system, a Scada system) and it is now difficult to share information between these different systems because of their complex and dynamic nature.

Information Technology has seen the emergence of standards for the last five years. Among those standards, some concern Object Oriented Techniques. These techniques promote a component-based client-server architecture that has the potential to overcome actual limitations. Object Oriented Analysis (OOA) will be used to develop the interface architecture and appropriate component interfaces. OOA is ideal for this task as objects include two fundamental characteristics:

- Abstraction: an object is described solely by its behaviour, expressed in terms of the published services it offers and uses. This means that implementation details are hidden.

- Encapsulation: information forming a part of an object can only be accessed through interfaces offered by the defined functions.

Electricity has seen an emergence of standards too. Amongst them some concern network automation, data acquisition and transmission. At the present time, there is a need to standardise applications, and inter-application communication, as the information technology market is mature and enough standardised. DMS technology is still in the development stage, not yet mature or stabilised. At this stage it could very likely be influenced by interface standards if these would ease its ability to inter-operate with other systems and applications. Through its distributed functions, a DMS plays a key role in the shaping of a utility's Information Technology (IT) architecture. It builds links between the various departments and their supporting systems. A certain standardisation of its interface to the other functional systems should minimise the implementation efforts.

The main objective of this work is to define standard interfaces that are implementation-independent and extensible, and thus guarantees their validity through future years.

Benefits to the Utility. From the point of view of the Utility, there are considerable advantages to be gained in acquiring systems that adhere to a recognised standard. These include:

- It will be easier to find vendors providing partial or total solutions for DMS applications. The optimum solution to match a particular utility's business process model may well involve component products from a number of different vendors.

- It will facilitate collaboration between utilities. In a deregulation market, it will help to have a common business process and will force utilities to minimise the development for their own specific business process. Utilities will focus more on the business process itself, knowing that the standard will cope with technological trends.

- If utilities still have a development division, they will become more integrators of component applications to map their business process. If they still have a specific business process, maybe it will be fulfilled partly by an off-the-shelf component application, making the adaptation easier and minimising the cost development of a total function.

- Utilities that develop high level functions could find more easily vendors or integrators, acting as partners, to integrate or to promote their function in a DMS solution.

- It will help Utilities to buy more off-the-shelf products as small suppliers could have access to the market so long as their product complies with the standard. The standard will act as a label of quality and compliance.

- The standard is a good communication tool. Even if a specific business process has to be included, it will be easier to integrate an application conforming to the standard. As the standard promotes Application Programming Interfaces, utilities and vendors/ integrators will focus more on this level than on the internal characteristics of an application.

- The existence of a standard interface to exchange information will allow the electrical utilities to choose the best supplier for every application from an economic and technical point of view. This circumstance will help the utility to increase quality of service and to offer energy at competitive prices.

- The DMS functionality may be expanded easily, because the system interfaces support the addition of new components instead of only complete, monolithic applications, without changing the installed base of applications.

- The interoperability of heterogeneous Distribution Management applications will be enhanced. Gradual addition or gradual replacement of Distribution Management applications will be feasible. The provision of standardised interfaces supports gradual migration because each function is viewed as a black box. This makes it possible for legacy (existing, monolithic) distribution management applications to integrate flawlessly with other applications, whether new or existing, because no requirements are placed on the implementation within the black box.

- Less risk in operating distribution management applications because proven interface technology is used.

- The costs of acquisition and maintenance will certainly decrease. Installation, training, and operating cost will be reduced because no external gateways or special interfacing software is needed.

- Contributing to the standard gives utilities an awareness of technological direction. Knowing that a technological solution exists helping to solve its business process, a utility will gain considerable benefit if it adheres to the standard.

Benefits to the Vendor. From the point of view of the Vendor, there are considerable advantages to be gained in designing and producing systems that adhere to a recognised standard. These include:

- It will be easier to find partners to work together to provide total solutions for DMS applications. A major DMS may encompass SCADA, mapping, outage management, network analysis, customer information systems, asset management, human and financial resource management and more. The optimum solution to match a particular utility's business process model may well involve component products from a number of different vendors.

- DMS projects should become shorter, more manageable and more profitable. In both the utility's and the vendor's interest, solutions which minimise development and therefore implementation risk by utilisation of proven products will be preferable.

- As a result of the above, some vendors will focus on integration, licensing component applications to build into their solutions. Others may prefer to become simply licensors of standard products, allowing others to take the integration risks.

- Vendors will have a more dynamic and attractive offer to present to the market. All vendors need orders and no doubt will waste no time in announcing compliant systems. Such claims will need to be weighed carefully.

- Small suppliers will be able to gain credibility. The implementation of a major DMS system is today beyond the capability and resources of a small supplier but, if he has a product that conforms to the standard, he will be able to select partners to provide a total DMS solution.

- The standard is a good communication tool, allowing both vendors and utilities to talk the same language and reducing the need for lengthy specifications. The possibility of misinterpretation of requirement scope is reduced. The costs of requirements capture, bid preparation, tender evaluation, work statement preparation and system analysis and design are all reduced.

- Contributing to the standard gives to vendors awareness of market direction. DMS is still a fledgling field with different needs and priorities evident in different countries. The new standard provides leadership and direction to this emerging market.

To this end, for a serious vendor or utility there is no alternative but to participate, since the market will demand products that conform to the standard.

IEC TC57 Working Group 14

Working Group 14 (WG14) was commissioned in September 1996 and has met three times each year since. Much of the work to date has encompassed problem definition, liaison with other parallel and related standardisation efforts both inside the IEC and outside and the selection of a suitable methodology. A significant input has been the research and analysis by the CIRED Working Group on Distribution Automation, published in 1996.

The Working Group includes amongst its membership 27 representatives from nine countries and from different disciplines including power utilities, specialist consultants, research institutions, power control system vendors and GIS vendors.

The first phase of the project, Parts 1 and 2 of the new standard, to be known as IEC 61968, has been completed and these parts have been passed to the various National Committees at the end of 1998 for their comment and input. Part 1 identifies and establishes requirements for standard interfaces based on an Interface Reference Model (IRM). Part 2 is the Glossary for the standard. Subsequent parts will define each interface identified in the IRM.

Co-ordination with other Standards Committees

The market place for Distribution Management Systems is still developing and Working Group 14 has been conscious of the need to co-ordinate their efforts with other standards committees and groups who are working in the DMS and related fields. There are several other such Working Groups operating under the aegis of IEC TC57. Their areas of responsibility are shown in Figure 1 below:

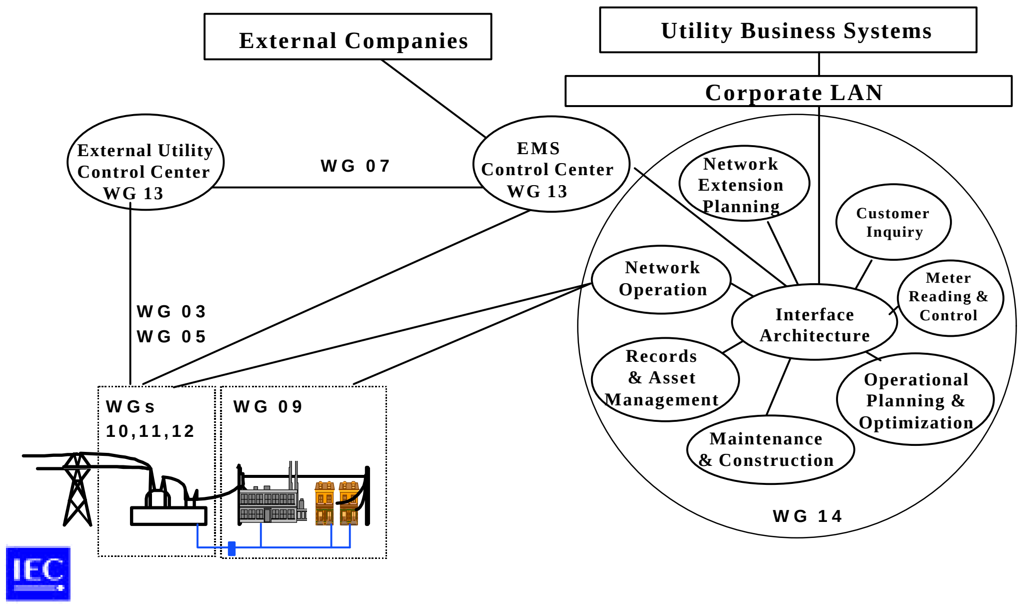


Figure 1: Working Groups within IEC TC57

The operations of the various TC57 Working Groups are co-ordinated by a Strategic Policy Advisory Group (SPAG). In addition WG14 monitors the work of various other standards-defining bodies and development projects where they are working in the Distribution Management field as described in Figure 2 below.

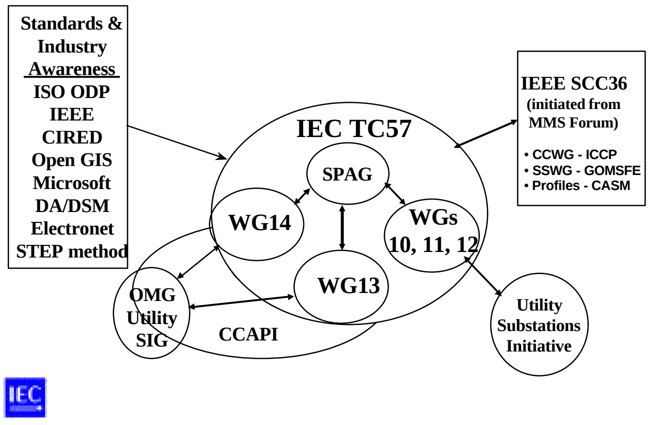


Figure 2: Co-ordinating amongst standards activities

Of particular significance to the work of WG14 has been:

- The report of the CIRED Working Group on Distribution Automation, already mentioned.

- The CCAPI project in the USA which, although initially primarily aimed at transmission and generation control systems, is being extended to encompass DMS systems. They have contributed the Common Information Model (CIM) to describe power networks and are establishing a message bus interface.

- The Utility Integration Bus (UIB) project which is being developed at Kansas City Power and Light.

- The work of TC57 Working Group 13 which is complementary to Working Group 14's, but for transmission and generation. They have adopted the CIM, introduced some refinements and have passed it to the National Committees for comment in October 1998.

The phases of development of the standard

Parts 1 and 2 of IEC 61968 cover the general requirements which compliant systems must support. The remaining parts of the standard address specific interfaces for the various business activity segments defined in the IRM:

Phase	IEC 61968 Part	Title
1	1.	Interface Architecture And General Requirements
	2.	Glossary
2	3.	Interface Standard For Network operation
	4.	Interface Standard For Records And Asset Management
Future	5.	Interface Standard For Operational Planning And Optimisation
	6.	Interface Standard For Maintenance And Construction
	7.	Interface Standard For Network Extension Planning
	8.	Interface Standard For Customer Inquiry
	9.	Interface Standard For Meter Reading And Control
	10.	Interface Standard For Systems External To, But Supportive Of, Distribution Management

TECHNOLOGY OVERVIEW

Methodology

IEC 61968-1 describes utility inter-application infrastructure requirements necessary to integrate components distributed throughout the enterprise. The services and functionality described is independent of the underlying component-based infrastructure. In the requirements, an "event" is a unit of information exchange that is issued asynchronously by its source ("push"). A "component" is a module of application software that is a component of the integration bus as either a publisher or subscriber (receiver) of an information exchange.

The business process begins by identifying the information to be exchanged and the components involved. This typically involves one publisher that has the information and initiates the exchange, and 0 to n subscribers that will receive the information.

IEC 61968 requires that a compliant utility inter-application infrastructure:

1. Shall allow components to exchange information of arbitrary complexity.

2. Shall be able to be implemented using various forms of distributed component technology (e.g., CORBA, DCOM, message brokers, message oriented middleware, relational databases, object-oriented databases, or others).

3. Shall provide an Information Exchange Model facility that users employ to describe the information to be exchanged.

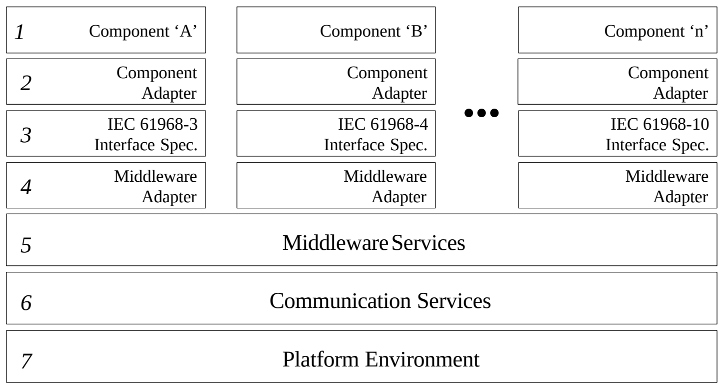
4. Shall allow publisher and/or subscriber components to be deployed by system administrators independently of other components.

5. Shall ensure that published information is completely re-usable in the sense that once a given type of event is published, any new authorised entity may acquire the event without having to make any changes or additions in the publisher component.

Requirements Analysis Methodology. To help solve the problem of effectively sharing information across Electric Utility Departments and systems, a common modelling notation or language is needed. A modelling language extends natural language by adding formal constructs to aid in communication by reducing ambiguity. By using a common modelling language across the utility, utilities can better define what information needs to be shared across departments.

IEC 61968 recommends that system interfaces of a compliant utility inter-application infrastructure be defined using Unified Modelling Language (UML). UML consists of modelling concepts and a defined notation. The modelling concepts are Use Cases, Objects and their relationships, Sequence and Activity Diagrams, and Component and Deployment diagrams. Use Cases are defined process for capturing system requirements and generating the information necessary to drive the other models. Static and Dynamic Object models show attributes, state changes and relationships (such as collaboration) of the problem domain objects. Sequence and Activity Diagrams show the sequence and flow of objects through a system. Component and Deployment diagram show the implementation of logical object models into real world code and architecture.

UML offers a large tool chest to analysts, designers and implementers. These tools help guide system and application designers in creating architectures and systems to meet stated requirements. Not all of UML tools are needed to meet WG14 objectives. However, with UML it is necessary to generate Use Cases because Use Cases provide the information necessary to build the other UML models. WG14 created a set of Use Cases describing how DMS users would like to share information between different DMS applications. From these Use Cases, WG 14 can identify common requirements that drive the design of both the Interface Architecture and specific DMS interface standards.



Overview of the Services Profile. The requirements for all the individual parts in this Service Profile are explained in the following paragraphs.

Information exchange among components can either be a piece of data or the result of an execution of functionality and for this purpose is called a services Exchange. For example, a Component can be a classic, procedural or a fully object-oriented application build around the latest technology. Also, Components can be distributed across the network enabling flexible deployment of DMS applications in the utility-wide IT-architecture. The scope of a Component is unlimited: it can perform any function that is required for Distribution Management.

A component can either be *Services-compliant*, meaning that it knows, understands and satisfies Services requirements or *non-Services-compliant*. A Non-services-compliant Component must be made compliant before it can fulfil its role on the Services.

For example, each vendor of today's DMS applications may have its own application architecture, its own API and its own mechanism of interfacing the application with other products of the same vendor. Such existing applications may very well have an important role as a client of the Services. But the industry cannot expect that a vendor rebuild all its existing applications to new versions that are Services-compliant. Even new applications may not always be Services-compliant, but instead use the established vendor-specific architecture and application interface. Therefore, *non-Services-compliant* Components probably will be in the majority during the early stages of the IEC 61968 standard series. When IEC 61968 becomes more widely accepted *Services-compliant* Components will become more widely available.

Component Adapters. A Component Adapter in the context of IEC 61968 is IEC 61968-compliant software that enables a non-compliant software application to use the services. As such, the component adapter only goes as far as necessary to make the Component conformant to one or more specific IEC 61986-3 interface specifications.

This implies that:

- For Components that already are services-compliant, the Component Adapter is not necessary.

- When a non-compliant Component is used in the services-environment, at least one Component Adapter is present for that Component to make it services-compliant. It can also be the case that more than one Component Adapter is used to make a single Component compliant with the services (e.g. one Component Adapter for each IEC 61968 interface specification).

- For those Components that are non-compliant, each Component Adapter is custom-made for that specific Component because it depends heavily on the architecture and implementation of the Component. A Component also runs in a specific hardware/operating system (HW/OS) environment. Therefore the triple set Component, (set of) Component Adapter(s) and HW/OS are fully dependent on each other.

How the Component Adapter makes a non-services-compliant Component compliant to the services, depends on the Component and the role it performs. A complication is that a Component that was not coded to be services-compliant cannot be made services-compliant directly, and that each Component is different.

Interface Specification. The IEC 61968 Interface Specification requirements consists of three parts: Component-specific specifications, requirements that refer to services specific for the utility domain and requirements that refer to services which are common in a distributed computing environment based on components. Individual IEC 61968 Interface specification for business activity areas (see above) are available in following parts of this IEC 61968 standard (IEC 61968-3 and further). For all three parts in an IEC 61968 Interface specification, it is required to:

1. be declarative, containing pre- and post-conditions, attributes, methods and parameters as needed for all the service exchanges that are part of the specific interface specification

2. be programming-language neutral

3. emphasise the separation of interface and implementation

4. be middleware-independent

Requirements for Component-specific Interface specifications are exclusive usage of IEC 61968 Interface services for those requirements that can be supported by those services. This means that for not-covered requirements, additional services may be specified (and probably need to be programmed or mapped in the Middleware Adapter or Middleware services). Required services for the DMS domains are:

1. Information Exchange Model Access service: This service allows distributed components to enter and discover the exchange syntax of registered components and be notified when changes occur

2. Directory Based services: These services allow access to, initial and runtime population of, and browsing of the available exchanges and services on the services. Different views based on specific properties can be defined in this service. For example, this allows the definition of a deployment property view for critical items that can be compared to the runtime property view to determine if these items are present on the Services. The Directory contains among other things, IDs for component and business objects templates (classes) as well as instances.

3. ID Factory: This service allows for the creation of a unique ID. There may be multiple types of ID's based on specific rules for each type

4. Persistent Exchange service with checkpoint facilities, allowing components that register at different times to synchronise status with other components. This service supports entering and purging exchanges, marking (a group of) exchanges and reading (a group of) historical exchanges

5. System Administration: This service interface allows administration and monitoring of exchanges and components on the Services. Component failure and load balancing are also part of this service.

6. Configuration: This service provides an interface for components to obtain their configuration from a persistent data store at start up or while running after a component has been partially configured locally.

7. Filtering: This service allows for the definition and applying of filters based on exchange types and contents.

Required common distributed computing services in Distribution Management are:

1. Component Life Cycle Service: allow the starting, stopping, and control of components to be executed on the Services.

2. Naming Service: This service provides a component naming service that supports a hierarchical structure and allows a component to locate other components using a human readable name. The naming service supports use of existing utility names, as well as creation, removal and aliasing of names.

3. Time Service: provides a way for distributed components to all have the same time with a configurable accuracy.

4. Concurrency Control Service: This service facilitates management of shared, similar items that are distributed on the Services, for example when multiple Components take care of different parts (exchanges, exchange types) of the same business object in real life.

5. Security services: allow an application to set and verify the privilege level of components and users with which exchanges are being performed, as well as encryption and decryption of individual exchanges. This service also supplies host authentication i.e. authentication of node(s) that attach to the Services.

6. Transactional Service: allows an application to declare the beginning and end of a multi-step transaction that either succeeds or fails as an atomic unit.

7. Component Interaction services: allow for reliable message transfer with a selectable Quality of Service. The Component Interaction services allow for life-cycle management of interaction services (create, delete, copy and move) and querying of established interaction (mainly valid for Publish and Subscribe interactions).

8. The Publish and Subscribe Messaging Service, which allows for synchronous and asynchronous message transfer between decoupled (anonymous) component instances.

9. The Request/Reply Messaging service, which allows for reliable synchronous message transfer between coupled, identified component instances.

10. The Publish/Reply Messaging service, which allows for a decoupled initiation of a message transfer (Publish), that is finished by a coupled transfer (Reply).

11. Workflow services: allow application level programmers to create and run business process automation agents.

Middleware Adapter. A Middleware Adapter in the IEC 61968 standard is IEC 61968-compliant software that augments existing middleware services so that the utility's inter-application infrastructure supports required Services. As such, the Middleware Adapter only goes as far as necessary to make the used set of Middleware services conformant to the requirements of one or more available IEC 61986-3 interface specifications. In this context the Middleware services represent not one single interface, it represents a set of interfaces to a set of corresponding services for Components.

Interface Reference Model (IRM)

A key part of the work of IEC 61968 Part 1 has been the decomposition of the DMS business domain into manageable components. From an early stage, the Working Group decided that this decomposition should be by business function rather than by the the domains of existing products in the market place, for example Geographical Information, Trouble Call and SCADA systems, since these in themselves varied widely in their coverage of requirements and overlapped significantly.

Two levels of decomposition were found to be necessary:

- Business activity segment which contains
- Component applications

Components are the lowest level for which an IEC 61968 interface will be defined. A single IEC 61968 specification will be issued for each business activity segment. It is envisaged that components within a segment may be supplied by different vendors and components may or may not use IEC 61968 services to communicate within a segment.

This decomposition is to some extent subjective since throughout the world utilities operate with different organisational structures. Nonetheless it is essential that the segments and components are easily recognisable to utilities.

Business activity segments

Various departments within a utility co-operate to perform the operation and management of a power distribution network; this activity is termed Distribution Management. Other departments within the organisation may support the Distribution Management function without having direct responsibility for the distribution network. This segmentation by business function is provided in the Interface Reference Model (IRM), which is described in Figure 3 below.

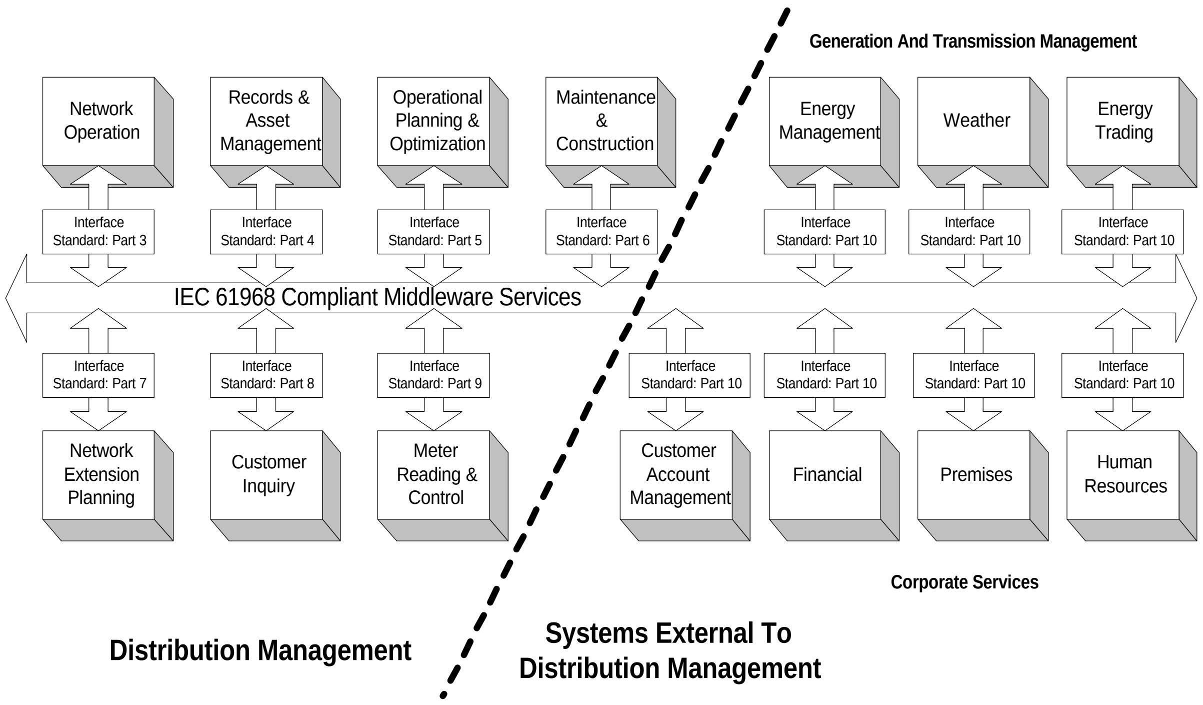

Generation And Transmission Management
Network Operation
Records & Asset Management
Operational Planning & Optimization
Maintenance & Construction
Energy Management
Weather
Energy Trading
Interface Standard: Part 3
Interface Standard: Part 4
Interface Standard: Part 5
Interface Standard: Part 6
Interface Standard: Part 10
IEC 61968 Compliant Middleware Services
Interface Standard: Part 7
Interface Standard: Part 8
Interface Standard: Part 9
Network Extension Planning
Customer Inquiry
Meter Reading & Control
Customer Account Management
Financial
Premises
Human Resources
Corporate Services
Distribution Management
Systems External To Distribution Management

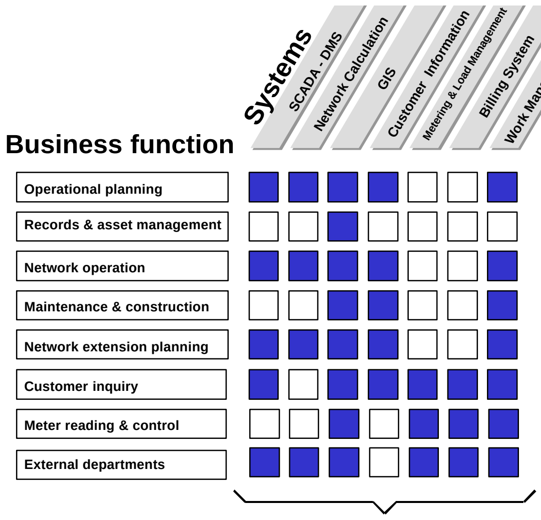
Figure 3: Interface Reference Model

The business activities that are directly involved in the operation and management of a distribution network are classified as follows:

Network operation	Switchgear operation, SCADA, fault management, feedback analysis, reporting, real-time calculations, protection, transformer load management, regulatory compliance, equipment diagnostics
Records and asset management	Substation and network inventory, geographic inventory, equipment analysis
Operational planning and optimisation	Load forecast, Contingency analysis, Short-circuit analysis, Optimal power flow, Switching simulation, Field crew loading analysis, Work scheduling, Outage analysis
Maintenance and construction	Periodic maintenance scheduling, Unscheduled work orders, Work order estimates, Maintenance crew management, Work state supervision, Materials inventory, Construction crew management
Network extension planning	Load forecast, Power flows, Contingency analysis, Short-circuit analysis, Optimal power flow, Energy loss calculations, Feeder voltage profiles, Construction costing, Work management
Customer enquiry	Customer information, Service orders, Outage reporting, Crew management, Outage management, Marketing
Meter reading and control	Load characteristics, Consumption meters, Quality factors, Load control, Dynamic tariff aplication, Power modulation, Customer outage detection, Remote service connection/disconnection.

As well as those business activities directly involved in the operation and management of the power network, other activities support these indirectly. These include customer account management, weather, financial, premises management, human resources, EMS and energy trading systems. A generic system interface specification (IEC 61968 Part 10) will be produced to cover these interfaces.

It would be true to say that the use of a business-related interface reference model has caused some friendly controversy within the working group. It is important to realise that particular systems do not necessarily fall into one business activity segment, but may span several as can be seen from the following matrix. This shows the mapping of typical utility systems to the business activities of the IRM.



Business function	SCADA - DMS	Network Calculation	GIS	Customer Information	Metering & Load Management	Billing System	Work Management
Operational planning	■	■	■	■	□	□	■
Records & asset management	□	□	■	□	□	□	□
Network operation	■	■	■	■	□	□	■
Maintenance & construction	□	□	■	■	□	□	■
Network extension planning	■	■	■	■	□	□	■
Customer inquiry	■	□	■	■	■	■	■
Meter reading & control	□	□	■	□	■	■	■
External departments	■	■	■	□	■	■	■

Figure 4: Function Oriented Information Technology

IRM Components

In this section the definitions of business activity segments defined above are further extended into:

- Components
- Functions within each component.

Business activity segment	Component	Function
Operational planning and optimisation	Network operation simulation	Load forecast
		Network calculation
		Supply restoration assessment
		Switching simulation
		Incident simulation
	Switch action scheduling / operation work scheduling	Release/clearance remote switch command scheduling
		Field crew loading analysis and work order scheduling
	Power import scheduling and optimisation	Customer outage analysis and information
Records and asset management	Substation and network inventory	Equipment characteristics
		Connectivity model
		Substation display
		Telecontrol database
	Geographical inventory	Network displays
		Cartographic maps
Network operation	Network operation monitoring	Substation state supervision
		Network state supervision
		Switching action supervision
		Management of data acquired from SCADA and metering systems
		Management of data acquired through operation (field crews, customers, scheduled and unscheduled outages)
		Alarm supervision
		Operator and event logs
	Network control	User access control
		Automatic controls:
		Protection (fault clearance)
		Sectionalising
		Local voltage/reactive power control
		Assisted control:
		Remote switch control
		Load shedding
		Voltage reduction broadcast
		Local control through field crews
		Safety document management
		Safety checking and interlocks
		Major incident co-ordination
	Fault Management	Trouble call handling and coherency analysis (LV network)
		Protective relays analysis
		Fault location by analysis of fault detectors and/or trouble call localisation
		Supply restoration assessment
		Customer incident information
	Operation feedback analysis	Incident simulation
		Network fault analysis
		Quality index analysis
		Device operation history
	Operation statistics and reporting	Maintenance information
		Information for planning
		Information for management control
	Network calculations – real-time	Load forecast
		Energy trading analysis
		Real-time network calculation
		Protective relays analysis
		Adaptive relay settings
	Dispatcher training	SCADA simulation

Business activity segment	Component	Function
Maintenance and construction	Maintenance works scheduling and control – power system	Periodic maintenance scheduling and unscheduled work orders
		Work order estimates
		Maintenance crew management
		Work state supervision
		Materials inventory
	Telecommunications network maintenance	Telemetry
		Communications links
		Servers/workstations/peripherals
	Construction work scheduling and control	Work scheduling
		Work order estimates
		Construction crew management
		Work state supervision
		Materials inventory
Network extension planning	Network calculations	Load forecast
		Network calculation studies
	Construction supervision	Construction costing
		Work management
Customer inquiry	Customer information	Address/network connection
		Customer data
		Interface to utility’s website
	Outage management	Outage reporting (Estimated-time-to-restore)
		Network fault analysis
	Crew management	Allocation/tracking
		Computer-aided dispatch
Meter reading and control	Meter reading	Load characteristics
		Consumption meters
		Quality factors
	Load control	Meter parameter telesetting
		Dynamic tariff application
		Power modulation
External to DMS	Energy Management (EMS)	Transmission
		Generation
	Weather	Forecast information
		Predicted dispatch of field crews
		Thermal ratings of major lines
		Lightning detection
		Fire risk
	Energy trading	Pricing
	Customer account management	Credit status
		Outage history
	Financial	Revenue information
		Costs
		Overhead
	Premises	Address
		Source substation
		Meter information
	Human resources	Health/safety reporting
		Staff credentials
		Hours on shift information

Middleware architecture and services

Distributed object technology aims at the integration of distributed, object-based systems. Components are independent software entities that encapsulate (private) data the component needs to know to perform its function. All component behaviour is defined by the component methods; and a user (client in a client-server relationship) can only access the component’s data by invoking these public methods. Components can reside anywhere within a single machine, or they may reside on other machines connected by the communication network.

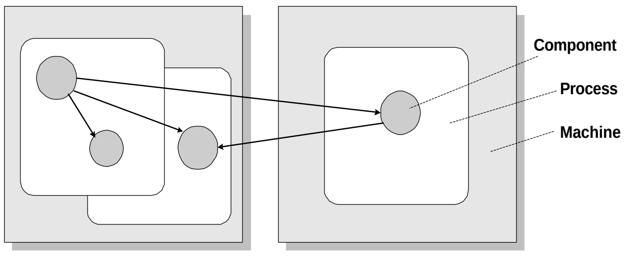


Figure 5: Distributed Components

Middleware is a term used to describe the software needed to support interactions between clients and servers. An interaction may start, for example, with an application on the client side that is used to invoke a service or method owned by an object. An application will use middleware to manage the transmission of the request over the network through the use of the communication services available from the selected Communication Service Provider (CSP). All communication actions and responses are enabled by the protocol of the selected CSP. Mainly, Middleware aims to make a heterogeneous, distributed environment appear as a single "virtual machine" that provides access to all resources and business components on the network, hiding the complexity of the necessary communication protocols and services. Middleware also hides the details of how the CSP implements the communication protocol.

Middleware has emerged as a strategic piece of industrial software and particularly of client/server systems.

The middleware market can be broken into the following general categories:

- **Database Middleware**: allows clients to invoke Database services across multi-vendor databases. This middleware is defined by de facto standards as ODBC, DRDA, RDA etc.

- **Transactional Middleware**: Software that allows clients to invoke services across multiple transaction servers.

- **Network Operating System Middleware**: NOS middleware extends the local operating system's reach to include networked devices. It provides a distributed computing environment that creates a " single system " out of the diverse resources distributed on the network.

- **Message Oriented Middleware**: MOMs allow general-purpose messages including procedures, data and controls to be exchanged between clients and servers via messages. MOM is a loosely coupled exchange across multiple operating system. Program-to-program communication via MOM is also referred to as "asynchronous".

- **RPC Middleware**: Enable a programmer to develop an application in which different procedures can be distributed and remotely invoked across one or more networked systems, yet appear as though they were executed on the user's system. These procedures are synchronous in nature.

- **Object Request Broker Middleware**: allows clients to invoke methods or components that reside on a local/ remote server. This middleware mainly revolves OMG's CORBA compliant ORB services and Microsoft's DCOM services. .

- **Middleware Adapters**: Middleware Interface Adapters may bridge CORBA compliant ORB-services from any vendor, DCOM- services, and application-specific ORB-services.

Information on the Component. Instead of building monolithic or two-tier applications, organizations (both vendors and utilities themselves) must break applications and subsystems into components. Components enable the utility and the vendor of application software to react quickly to changes in the business environment, allowing them to update only the parts of a DMS application that require it. In this context a "Component" may be a classic, procedural or a fully object-oriented application build on top of the latest technology.

These components can be distributed across the network (LAN, WAN and Internet), enabling flexible deployment of DMS applications in the utility-wide ICT-architecture.

Information on the Component Adapter. Proprietary, vendor-dependent solutions hinder the interoperability on which distributed (object) computing is based if deployed directly in the utility-wide ICT-architecture. However, each vendor of today's DMS applications has its own application architecture, its own API and different mechanisms of interfacing the application with other products of the same vendor.

To create cross-platform and cross-vendor utility solutions, it is necessary to adopt clear interface specifications for well-defined Components enabling the (ideally) "hot plug-in" of additional components without altering the ICT-architecture or changing applications. To be able to use these standard interfaces, it is necessary to bridge the gap between the standard interface on one side and the vendor-specific way of interfacing its own Component to the outside world on the other side. The component adapter is the part in WG14's Interface Architecture that bridges that gap. The component adapter establishes the "component view" of the application (component) it "shields": it translates whatever is behind it to an Object-Oriented view.

In the three following paragraphs, we expose the characteristics of two well-known middleware environments in the IT market, CORBA and DCOM. The third paragraph tries to make a brief comparison between the two.

CORBA (Component Object Request Broker Architecture). CORBA is a de facto-standard promoted by Object Management Group, which is an international organisation supported by over than 750 members. The OMG promotes the theory and practice of object-oriented technology in software development. OMG established the Object Management Architecture (OMA) which provides the conceptual infrastructure upon which all OMG specifications are based.

The basic components of a CORBA-compliant ORB are as follows:

- **ORB Core**: The ORB core is the object bus that provides the middleware that mediates the interactions transparently between client and server applications

- **Client Stubs**: Client stubs are static bindings that will provide access to the IDL-defined operations of an object in a programming language the IDL definition has been mapped.

- **Dynamic Invocation Interface**: Through the Dynamic Invocation Interface, clients who do not have any static bindings to the server object can construct requests on the fly. Clients literally compose the request at run time and dispatch it to the server

- **ORB Interface**: The ORB interface goes directly to the ORB, which is the same for all ORBs. It provides interfaces that are common to all objects and therefore are useful for both clients and implementations of objects.

- **Implementation Skeleton**: The implementation skeleton is the static interface through which the ORB calls the methods that implement each type of object.

- **Dynamic Skeleton Interface**: In CORBA 2.0, the dynamic skeleton interface (DSI) is the server-side analogue of the dynamic invocation interface. It is part of the bridging mechanism between different ORBs.

- **Object Adapter**: It handles services such as the generation and interpretation of object references, method invocation, security of interactions, object and implementation activation and deactivation and the registration of implementations.

- **Interface Repository (IR)**: By querying the IR at run-time, it is possible for a client to encounter an object whose interface was not known at compile-time. Thus the client is able to determine what operations are valid for the object and then make an invocation on it.

- **Implementation Repository**: The implementation repository contains information that allows the ORB to locate and activate object implementations.

Providing application invocations over a network is not sufficient for distributed computing. With respect to distribution services, the CORBA services offer a set of services assembled in one distributed computing specification, including DCE and DCOM.

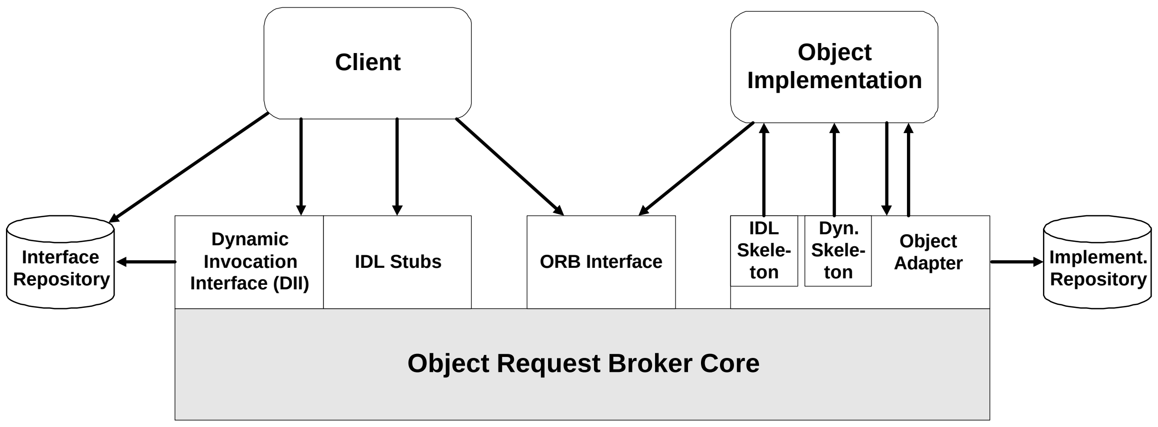


Figure 6: Basic Components of a CORBA-compliant ORB

Table 1 : CORBA Object Services

Service	Definition
Lifecycle	Manages the creation and destruction of objects.
Naming	Manages the naming of objects within defined naming contexts.
Events	Manages channelling of events between event producers and consumers.
Persistence	Enables objects to persist beyond the completion of the process that creates them.
Externalization	Supports export and import of object state information.
Relationships	Manages bi-directional associations between objects.
Transaction	Manages atomic units of work involving operation invocations.
Concurrency	Manages concurrent access to shared objects.
Security	Defines interfaces for authentication and authorization of operations on objects.
Time	Synchronizes clocks in a distributed system
Licensing	Manages licensing agreements between objects.
Properties	Manages dynamic named attributes associated with objects.
Query	Provides SQL-like access to sets of objects.
Collections	Supports groupings of objects
Trading	Enables clients and servers to match services with needs.
Startup Service	Supports sequences of requests issued on ORB start-up and shutdown to establish predictable server configurations on start-up and restart.
Change Management	Provides versioning and composition of objects that change over time.

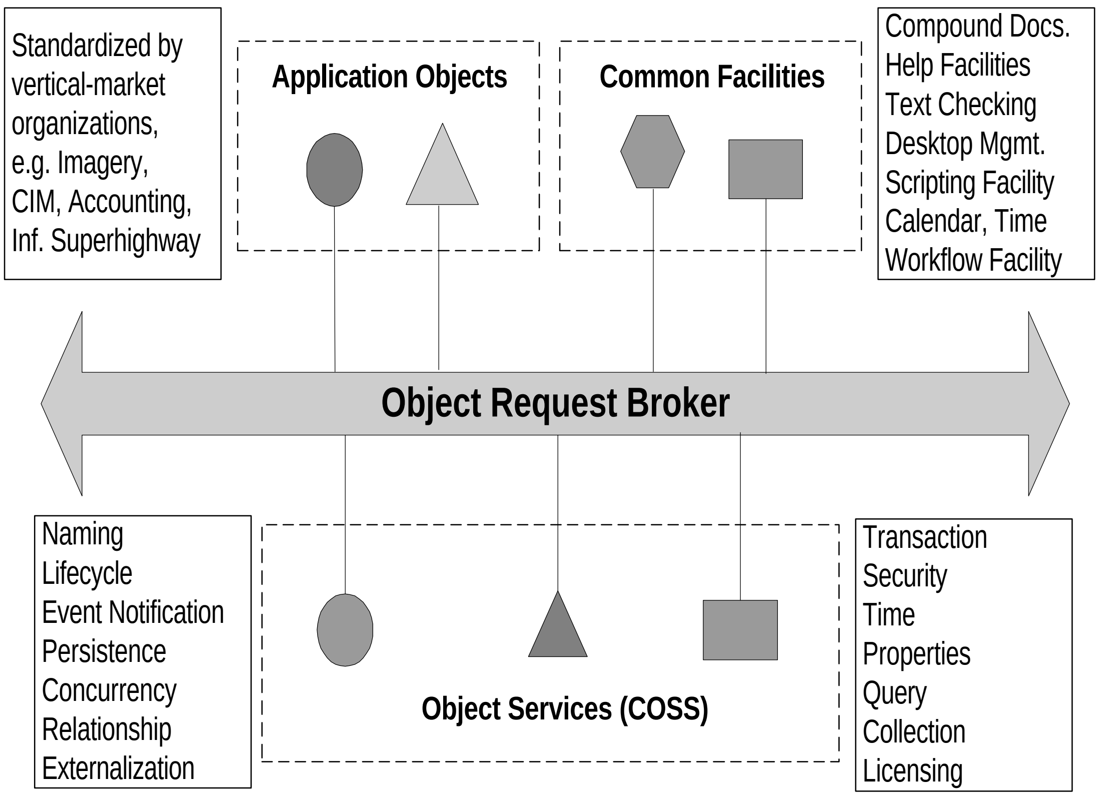


Figure 7: Object Management Architecture

The main services of the CORBA-compliant Object Adapter are listed below and it applies the numbering scheme of Figure 8:

- **Register Server Classes** with the Implementation Repository. Object Implementation classes are registered a and stored as a persistent store in this repository.

- **Instantiate new objects** at run time. The object adapter is responsible for creating object instances from the implemented classes. The number of instances is dependent of the incoming client traffic load.

- **Generate and manage object references**. The object adapter assigns references (unique Identifiers) to the new Components objects it creates. The Object adapter is responsible for mapping between the implementation- specific and ORB- specific representations of object references.

- **Broadcast the presence of the object servers**. The object adapter may broadcast the services it provides on the ORB and it may respond to directory type queries from the ORB- core. It is in charge to let the outside world know of the services it manages.

- **Handles incoming client calls**. The object adapter interacts with the top layer of the ORB core communication stack, peels off the request and hands it to the interface stub. The stub is responsible for interpreting the incoming parameters and presenting them in a form that's acceptable for the object's method invocation.

- **Routes the up- call to the appropriate method**. The object adapter is implicitly involved in the invocation of the methods described in the stubs (or the skeleton), e.g. the object adapter may be involved in activating the implementation and it can authenticate the incoming requests.

The **Interface Definition Language (IDL) is part of the CORBA** standard and is used for the definition of the interfaces of CORBA compliant objects.

An IDL compiler translates this IDL files in C++ header and source code files. The header file, called IDL C++ header, is used both by the client and server part of the application. The source code files contain the stub and skeleton code that must be compiled and linked with the client and server application respectively.

DCOM (Distributed Component Object Model).

DCOM is based on Microsoft’s Component Object Model (COM). COM is a binary standard, which means that COM interfaces are defined by the layout of virtual function pointer tables (vtables), in memory. There is also a standard way how functions are called through the vtables. Thus, any language that can call functions via pointers (e.g. C, C++, Smalltalk, Java, and even Visual Basic) all can be used to write components that can interoperate with other components written to the same binary standard.

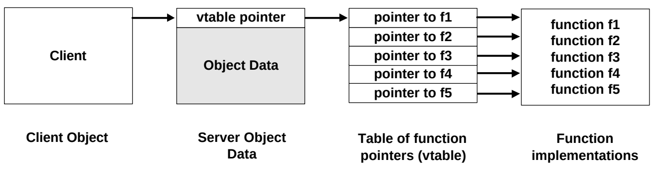


Figure 8: Binary COM Interface

Each entry in the vtable contains the address of a function implemented by the Component. In COM notation, this vtable is called interface. In an analogous way, the vtable pointer is call interface pointer. A client can communicate with the COM application component only through an interface. The vtable and the function implementations are shared among multiple instances of the same object class which reduces memory requirements.

COM preserves completely encapsulation of data and processing. When a client has access to a COM component, it has nothing more than an interface pointer through which it can access the functions of the interface. Therefore, client objects will never have direct access to the server object's data. This encapsulation is what allows COM to provide a binary standard that allows local and remote transparency.

The Component Object Model (COM) provides an inherent means (within the Windows operating system) for inter-applications communication. The basic COM services enable objects with standard interfaces to communicate, assuming that each object or is operating within the same process space, or to put it more simply, the same computer. COM uses a *proxy/stub methodology* for marshalling communications between client and server processes.

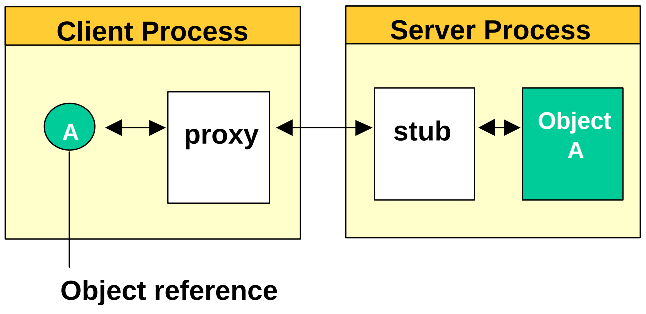


Figure 9: The Proxy/Stub Methodology of COM

DCOM adds three elements for remote server processes:

- techniques for creating a remote object(s)
- protocols for invoking that object's methods
- mechanisms to ensure secure access

At a high level, DCOM accomplishes the remote server processes through adding Network Transport services (NTS) to the COM proxy/stub.

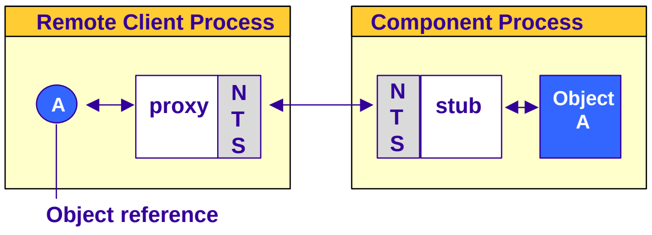


Figure 10: The Proxy/Stub Methodology of COM

DCOM uses a *system registry* to map the names of remote server objects, and the machines or process spaces on which they are located. Each registry is updated with a client-specified unique ID when an object is created. DCOM supports a variety of communications protocols for identifying remote machines, depending on the network protocol in use.

Network Transport services are actually provided in DCOM through the same mechanism as in CORBA, that is, through Remote Procedure Calls (RPCs). The specific RPC protocol used in based on OSF DCE (Distributed Computing Environment).

OLE comes with a bunch of services that are implemented on COM. Each of these technologies is defined by a set of COM interfaces. Every interface specifies a set of semantically related functions.

- Structured Storage and Persistent Objects for storing components of different types in the same file.

- Uniform Data Transfer defines a standard way for data transfer between components and applications. It provides the functionality to represent different kinds of data transfers through a single interface. Furthermore, it defines a simple mechanism of change notification.

- Naming and Binding services (Monikers) for the creation and initialization of COM objects

- Connectable Objects, a kind of event service that makes it easy for clients to receive notifications of interesting events

- OLE Automation refers to the ability of an application to define a set of properties and commands and make them accessible to other applications to enable programmability. OLE Automation is treated in more detail below

- Compound document services such as embedding and linking and in-place activation. These must be used by both the client (container) and server objects

- services for controls. The arrows indicate that controls depend from almost all other technologies.

Microsoft's Interface Description Language (MIDL) is based on the RPC IDL that is used in the Distributed Computing Environment (DCE) of the OSF. MIDL is used for the description of interfaces, operations, and attributes to define remote procedure calls. MIDL was tailored to support object-oriented method calls. The main purposes MIDL is the generation of proxy and stub code necessary for local and remote communication and the generation of type libraries that are used by OLE Automation clients and servers.

OLE Automation is COM's support for dynamic interfaces. It provides a particular interface called dispatch (or IDispatch) interface that allows applications to expose their services for clients written in simple languages such as Visual Basic. Applications providing a dispatch interface are sometimes called programmable applications. Examples for such applications are Microsoft's office applications such as Excel and Word.

Information on Middleware Interface Adapter. The COM / CORBA Interworking Specification from the Object Management Group (OMG) is a standard for interoperability between COM and CORBA. Products that implement this standard allow CORBA objects to be used from COM and vice versa. There are two parts to the interworking specification. The first part specifies OLE Automation-to-CORBA interoperability and the second part specifies COM-to-CORBA interoperability.

The two levels are mapping (one-way interoperability) and interworking (two-way interoperability). A mapping solution makes an object from one system available to the other system (e.g., CORBA objects are available to COM) but not vice versa. An interworking solution makes objects from one system and vice versa (e.g., CORBA objects are available to COM and COM objects are available to CORBA).

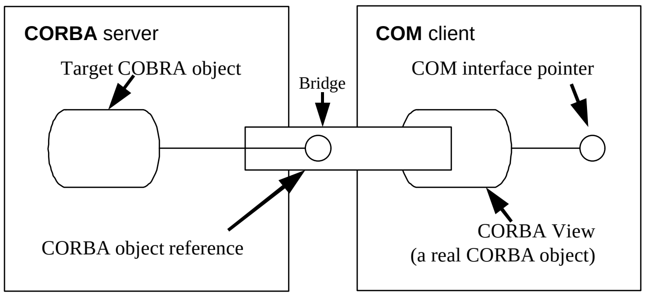


Figure 11: CORBA/COM Interworking Architecture

Differences between CORBA and DCOM

Despite some similarities, there are major differences between CORBA and COM in the way they implement their interfaces. COM specifies a series of interfaces that components must implement to interact with other component objects. All these interfaces must derive from a base interface. CORBA does not specify a single base class, and vendors get to implement their own.

CORBA is a specification without a reference implementation. Its lack of implementation detail is both its greatest strength and most severe limitation. While providing the flexibility for more creative vendor implementations, it also opens up issues such as inconsistent administration, incompatible ORBs, and non-portable servers. Microsoft COM, on the other hand, is very specific in its implementation detail, but this tight control can lead to non optimal solutions, especially in terms of performance.

COM is expected to become an industry standard in the desktop market. And many vendors are working to ensure that COM and CORBA will be interoperable. There is little doubt that the Windows NT Operating System will continue to grow as a replacement for Unix systems. If this becomes a widespread reality, with major hardware vendors porting and supporting Windows NT, Distributed COM is bound to play a prominent role in the Distributed interoperable objects arena.

Information Exchange Model

IEC 61968 Part 1 mandates that any compliant system shall include an Information Exchange Model (IEM). This may be physically distributed, but there can be only one logical IEM within a system.

The IEM contains descriptions (metadata) of the contents, syntax and semantics of information to be exchanged between components. This is sometimes referred to as a data dictionary. This information must be held in a publicly accessible, yet secure, manner.

It is important to understand that we do not advocate a centralised database; on the contrary, data is best held close to the component application where it is used. The purpose of the IEM is to facilitate the exchange of information between discrete databases associated with each component application.

Information is exchanged between components by the use of one or more events, whose types are defined in the IEM. The IEM contains names of:

- primitive data types,
- business object types (e.g. Breaker, outage schedule),
- names and datatypes of attributes of business objects (e.g. "in-service", "live"),
- relationships between business objects (e.g. "owned by", "connected to"),
- named event types which act on objects (e.g. Object creation/deletion, attribute update).

Event messages are managed on a "publish and subscribe" basis. A component can register itself as a publisher or subscriber of a particular type of event message. In addition, the component can register the context, real-time, study or test, for which it subscribes to or publishes a particular event type.

Maintenance and support of the IEM is also defined. It must be possible to:

- allow dynamic changes to the model without wholesale disruption to operation.
- validate its completeness and accuracy where possible (e.g. enforcing the uniqueness of names).
- synchronisation of components over updates using version control.

Event history, security, error reporting

IEC 61968 Part 1 also specifies the ground rules which ensure that compliant systems are maintainable and secure in their operation. This is done by specifying:

- an event history component
- security and authentication features
- error message handling

A compliant system must provide a generic **event history** facility as a component, where all or selected information exchanges are saved in a persistent store. A major benefit is to allow the taking down and bringing up of a component application without taking down the remaining components. Events for the affected application are recovered from the event history for the interim when the component was unavailable. Requirements for the event history include:

- The event history's schema is based on the metadata held in the IEM.
- Each event is individually time-stamped.
- IEM versions and component versions are supported.
- The event history component includes an Inter-application supervisor which enables the analysis of any of the application component interfaces. This can be enabled and disabled and is provided to give visibility to the performance of specific interfaces, so that bottlenecks can be identified and system availability ensured.
- The event history allows transparency to event-publishing components as to the availability of its subscribers and allows this to be managed externally to individual components.

A compliant system must also support **security and authentication** features. This covers both management of the integrity of data exchanged between components and also protection from accidental or intentional unauthorised access. Security functions are provided in two ways:

- Each component is responsible for ensuring that its users have the necessary authentication in order to perform a designated function.
- In addition a security agent is specified with responsibility for enforcement of authentication, encryption, access control and maintenance of security across the system as a whole. There will be only one security agent within a system.

Within IEC 61968 Part 1 only general statements can be made about **error-handling** since generally the requirements are specific to each component interface. Nonetheless it is required that:

- Each error report must contain sufficient information to make the report useful!
- There shall be different types of errors: warnings, non-fatal errors and fatal errors.

NEXT STEPS

New Work Item Proposals

After IEC 61968 Parts 1 and 2, the next stages are:

Part 3:	Interface standard for Network operations
Part 4:	Interface standard for Records and Asset Management

The Working Group is currently putting together new work item proposals covering this work for authorisation by the IEC. It is envisaged that the two specifications would be developed in parallel using the methodology described above, that is:

- The development of detailed use cases
- Static and dynamic object definitions
- Sequence and activity diagrams
- Component and deployment diagrams
- Inter-component infrastructure definitions in UML
- The development of the Interface Exchange model
- Development of prototypes of the interface

Funding needs

It will be appreciated that the research, analysis and hard work associated with the work of WG14 is very large. So far it has been undertaken by "volunteers". The next stages described in the paragraph above will require prototypes to be developed to ascertain the feasibility and practicality of the component interfaces that will be defined.

The Working Group would be interested to hear from any partnerships of utilities, research associations and vendors who would be interested in participating in these developments.

New CIRED working group

Historically, the CIRED WG 02 "Distribution Automation" analysed the business applications and produced recommendations to group them in activity-based segments.

The IEC WG14 "System Interfaces for Distribution Management" started from the CIRED WG02 final report. As this group is mainly composed of DMS manufacturers, it was very important to have a better feed-back from Utilities on WG14 work, and standard elaboration. As a consequence it has been decided to promote the creation of a new CIRED working group. Therefore a proposal has been made at the CIRED level at the end of 1998. This group should be created in 1999.

This new group will certainly focus on the following items, as they have been mentioned in the proposal :

- State of the art concerning the interoperability of Distribution Management Applications: the point of view of the utilities.
- Database management: practices and tendencies. Proposals for optimal database sharing between DMS applications.
- Interest and applications of the Distribution Management Interface Architecture proposed by the IEC WG14.

The final report of this group should be very valuable for the cause of interoperability in DMS, as it will reflect the views of the Distribution Utilities that are members of CIRED.

The creation of this group will reinforce the elaboration of the standard and it will help in delivering a standard which satisfies both vendors' and utilities' needs.