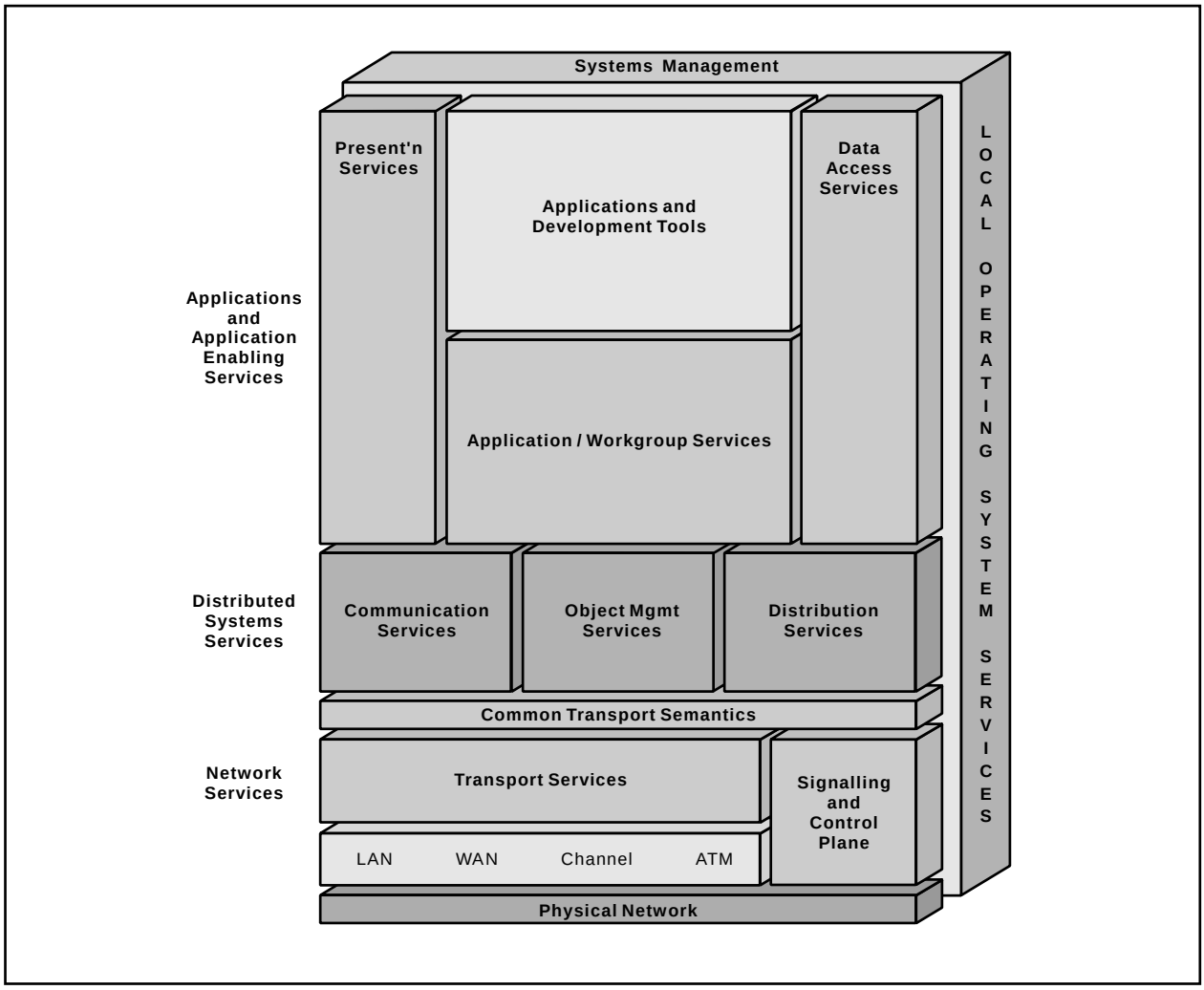
Open Blueprint

Compound Document Resource Manager



GC23-3913-01

Open Blueprint

Compound Document Resource Manager

GC23-3913-01

About This Paper

Open, distributed computing of all forms, including client/server and network computing, is the model that is driving the rapid evolution of information technology today. The Open Blueprint structure is IBM's industry-leading architectural framework for distributed computing in a multivendor, heterogeneous environment. This paper describes the Compound Document resource manager component of the Open Blueprint and its relationships with other Open Blueprint components.

The Open Blueprint structure continues to accommodate advances in technology and incorporate emerging standards and protocols as information technology needs and capabilities evolve. For example, the structure now incorporates digital library, object-oriented and mobile technologies, and support for internet-enabled applications. Thus, this document is a snapshot at a particular point in time. The Open Blueprint structure will continue to evolve as new technologies emerge.

This paper is one in a series of papers available in the *Open Blueprint Technical Reference Library* collection, SBOF-8702 (hardcopy) or SK2T-2478 (CD-ROM). The intent of this technical library is to provide detailed information about each Open Blueprint component. The authors of these papers are the developers and designers directly responsible for the components, so you might observe differences in style, scope, and format between this paper and others.

Readers who are less familiar with a particular component can refer to the referenced materials to gain basic background knowledge not included in the papers. For a general technical overview of the Open Blueprint, see the *Open Blueprint Technical Overview*, GC23-3808.

Who Should Read This Paper

This paper is intended for audiences requiring technical detail about the Compound Document Resource Manager in the Open Blueprint. These include:

- Customers who are planning technology or architecture investments
- Software vendors who are developing products to interoperate with other products that support the Open Blueprint
- Consultants and service providers who offer integration services to customers

Contents

Summary of Changes 1

Open Blueprint Compound Document Resource Manager 3
Functional Description 3
Relationships to Other Resource Managers 5
Interoperability Considerations 5
Application Development Support 6
Associated Support 6

Notices 7
Trademarks 7

Communicating Your Comments to IBM 9

Summary of Changes

This revision includes:

- Additional information about scripting
- Clarification about how remote communication fits with compound documents
- Integration of the Taligent CommonPoint technology into IBM OpenClass
- Editorial clarifications

Open Blueprint Compound Document Resource Manager

Compound documents refer to a new kind of component-based application that is an attractive alternative to monolithic applications. The Compound Document resource manager supports a component-based programming model in which small task-specific software components from multiple vendors can be assembled rapidly by programmers or end-users to create customized software solutions for a wide variety of business or personal tasks. Most software needs some way of allowing user interaction and presenting data or results. These presentations are considered part of a compound document, even though that document could be nothing more than a control panel, clock, or stock market ticker tape on the computer screen.

The term *compound document* is common in the industry, because this component technology was first applied to mixing data types and their editors within a single word-processing application. However, the technology is general in its ability to allow components to be integrated for data sharing, data interchange, presentation, and user interaction. Component function is not limited to data editing or entry. Components can update files, monitor physical equipment, and perform computations and perform other functions.

In a compound document, a distinction is sometimes made between a compound document component's data (called the *part*), and the libraries of code (called *part handlers*) that can understand, present, and manipulate that data. When this distinction is unimportant, compound document components are often referred to as parts. Parts and their associated part handlers can perform basic tasks such as text editing or charting, or can address more complex domain-specific tasks such as inventory control or account management. More complex parts will often be built from more basic parts. For example, an account management part would probably use a text editing part for presenting data and receiving new data entered into text forms. Building complex parts from more basic parts allows end-users to easily customize their solutions by substituting alternative parts.

The Compound Document resource manager supports a collection of object-oriented interfaces and protocols, which enable it to support the execution of software that has been developed using a component-based programming model. The fundamental role of the Compound Document resource manager is to invoke part handlers and orchestrate their interactions.

Functional Description

The major elements of the Compound Document resource manager are:

- Presentation management protocols
- Data interchange format and protocols
- Scripting enablement
- Modeling framework
- Presentation framework

In this paper, protocols consist of a set of object interfaces and the semantics of their proper use and interaction.

Presentation Management Protocols

Compound document presentation management protocols[1] conform to the Component Integration Laboratories (CIL)[2] OpenDoc specification, and ensure that components can be used together in a common presentation to the user. These protocols cover geometry management, human interface event distribution, shared user interface controls, and rendering management. Geometry management is provided to allow components to draw in the appropriate portion of the document's screen area. Human interface event distribution delivers mouse and keyboard input to the appropriate component within the document. Protocols for sharing user interface controls such as menus, palettes, and button bars allow a compound document to present a well-integrated human interface to the end-user. Rendering management ensures that components redraw their presentations as necessary to allow repair of damaged areas and printing of the entire document.

Data Interchange

Compound document data interchange conforms to the CIL OpenDoc specification, and includes a canonical storage format and protocols for storage management and data links between parts.

Bento, a canonical storage format, is supported to enable compound documents to be archived and transferred between machines with different architectures. Storage management protocols allow runtime access to component data (that is, the part as distinct from the part handler). These protocols provide a consistent way of accessing part information such as type, annotations, and associated presentation code (for example, part handlers). Annotations are arbitrary additional information associated with a part.

Parts can contain links to other parts, or can be the target of links from other parts. Links allow information to be passed between parts. There can be multiple part handlers that can handle presentation of a part. A part includes information that allows part handlers to be located and loaded into a document process. A component's part can include multiple stored data streams for that part data, each in a different format, or type. Part access and conversion protocols support exchange of part data in the desired type of format. For example, a text part might support both flat ASCII and Encapsulated PostScript (EPS) data streams. When that part is accessed, the text data can be retrieved in either format. Additionally, when a part does not store the desired format, the translation service is used to see if there is an available filter that can produce the desired type from the part type. Any vendor can supply additional translation filters to convert between part types. In our previous example, even though the part stores only ASCII and EPS data formats, the part data could be retrieved in Rich Text Format (RTF) if a part converter is available to transform EPS to RTF. This support is often used by clipboard cut, copy, and paste operations and by drag and drop embedding operations.

Scripting Enablement

It is becoming increasingly important to support rapid development using scripting or 4GL languages. The CIL OpenDoc Open Scripting Architecture (OSA) defines the scripting standard for compound documents. The Open Blueprint Compound Document resource manager will support OSA over time. OSA support includes both the use of Event Suites of standardized semantic operations and a direct scripting capability to get or set properties, invoke methods, or raise events on the compound document components.

An Event Suite is a set of actions to which a compound document component responds. OSA defines a base Event Suite that all compound document components must respond to. CIL provides an administrative procedure for registering additional, more specialized Event Suites that components choose to support.

Direct scripting provides a mechanism for a compound document component to support invocations from a scripting language that are not necessarily defined within an Event Suite. This approach is best suited for components that are developed together to form an application.

Modeling Framework

Within the Compound Document resource manager, a modeling framework is provided to assist in the creation of part handlers. The compound document modeling framework separates a part handler's function into model, command, selection, and presentation objects. Base classes for these objects and supporting classes hide many of the data interchange, presentation management, and scripting protocols from the developer of a part handler. Default part handler behavior is implemented by the framework for storage and retrieval of part data including embedded parts, linking between parts, unlimited undo and redo of model behaviors, and support for scripting through semantic events and object specifiers. Using this framework to create parts handlers ensures that parts will be well-behaved and have an extensible and flexible structure.

Presentation Framework

The presentation framework supports the creation of user interfaces (views) associated with a part's model, as implemented using the compound document modeling framework. These views are implemented using the Human Computer Interaction resource manager. It hides aspects of the presentation management protocols associated with keeping track of dialogs, menus, palettes, and controls needed for each component of a compound document.

Relationships to Other Resource Managers

Compound document support is implemented entirely as object-oriented frameworks and class libraries. These frameworks and libraries use System Object Model (SOM), as supported by the Object Request Broker resource manager, as the underlying object model. A compound document component, through its model implementation, can use the Open Blueprint Object Management Services' Object Request Broker (Distributed SOM) for remote access to objects. Remote data or functions not encapsulated in objects can be accessed using the Open Blueprint Communications Services. Compound documents are stored in files and directories, and therefore can be accessed and secured through the Open Blueprint Directory and Security Services.

Locally, the Compound Document resource manager makes extensive use of operating system services, including dynamic loading, shared memory, threads, and exceptions. The Compound Document resource manager is designed to work with IBM's complete set of C++ application frameworks, including both presentation management-based class libraries and IBM OpenClass class libraries, which provide interfaces to the Human Computer Interaction resource manager. Generally, parts can access all Open Blueprint interfaces. These application frameworks are provided on many different platforms. This makes it possible to create portable document components.

Interoperability Considerations

The compound document architecture and implementation supports interoperability with any OpenDoc part or IBM OpenClass document component. On Windows NT and Windows 95, Object Linking and Embedding (OLE) parts are also supported. Such parts can be embedded in OpenDoc containers on the same machine through drag/drop or cut/copy/paste mechanisms. They can also be actively linked into OpenDoc parts. Likewise, OpenDoc parts can participate in OLE containers. Data can be transferred in both directions between OLE and OpenDoc parts or containers. OLE parts can be scripted with OpenDoc

parts. The extension of interoperability support to include Java Beans and ActiveX components is under consideration.

Compound documents can be stored in Bento, which provides a canonical format for the storage of compound document content and which allows shipment across machine architectures.

Parts are SOM objects which can be designed using the model, command, selection, and presentation paradigm. The compound document modeling framework can be used to create structured parts. The model portion of such a part can be distributed using the Object Request Broker resource manager, provided that the implementation meets the restrictions of the Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) data types.

LotusScript is supported for scripting compound documents, with other scripting languages to be supported in the future.

Application Development Support

Document components are developed as SOM objects. Extensive C++ class libraries make component development highly productive, and enable creation of part handlers that are portable across operating systems. These class libraries include Taligent CommonPoint function. The functions in these libraries are categorized as either application services or application structure frameworks.

Application services can be used to incrementally enhance existing applications, turning them into part handlers and improving their portability across multiple platforms. Application structure frameworks provide basic application structure, and therefore are applicable to new application development or rewriting of existing applications.

Application services include graphics rendering, printing, collections, internationalized text, notification, database access, tasks, files, semaphores, timers, and many other classes. Application structure frameworks include the compound document modeling and presentation frameworks described previously, and basic user interface frameworks and higher-level part frameworks such as a form part framework. For some functions, notably user interface, there is a choice of class libraries to suit different functional or machine requirements.

Associated Support

As component technology becomes more widely used, a set of standard parts on all platforms for basic document exchange, and for building more complex parts from these core parts will be supported. IBM plans to offer a rich set of parts across the major client and server platforms, and is working with other companies to agree on standard part types and data formats.

---

1 While the Compound Document resource manager “controls” presentation through functions, it relies on the Human Computer Interaction resource manager in the Presentation Services portion of the Open Blueprint to construct the presented view.

2 Component Integration Laboratories (CIL) is a consortium that provides a set of open technologies that support the integration of applications.

Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

IBM
IBMLink
Open Blueprint
OpenClass
Presentation Manager
System Object Model
SOM

The following terms are trademarks of other companies:

C++	American Telephone and Telegraph Company, Incorporated
CORBA	Object Management Group
Encapsulated PostScript	Adobe Systems, Incorporated
EPS	Adobe Systems, Incorporated
OpenDoc	Apple Computer, Incorporated
Taligent	Taligent, Incorporated
Windows NT	Microsoft Corporation

Microsoft, Windows and the Windows 95 logo are trademarks or registered trademarks of Microsoft Corporation.

Communicating Your Comments to IBM

If you especially like or dislike anything about this paper, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply. Feel free to comment on specific error or omissions, accuracy, organization, subject matter, or completeness of this paper.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

- If you prefer to send comments by FAX, use this number:

  United States and Canada: 1-800-227-5088.

- If you prefer to send comments electronically, use one of these ID's:
  - Internet: **USIB2HPD@VNET.IBM.COM**
  - IBM Mail Exchange: **USIB2HPD at IBMMAIL**
  - IBMLink: **CIBMORCF at RALVM13**

Make sure to include the following in your note:

- Title of this paper
- Page number or topic to which your comment applies

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC23-3913-01