A FINAL FORM DOCUMENT ARCHITECTURE FOR INTEGRATING MULTIPLE DATA FORMATS

Reinhard H. Hohensee*

Keywords: Color, Document, Format, Information, Print.

Abstract: The Mixed Object Document Content Architecture (MO:DCA) is used in IBM presentation environments to define an interchange format for final form documents. The architecture defines document content and layout in a hierarchy of lower-level components and objects using a natural, self-defining syntax.

Traditionally, pages in MO:DCA documents were built with data objects that were limited to a small number of fixed formats. An extension is being developed that will allow pages to be built with a mixture of data specified in any of the popular data formats used in the graphics industry. This extension proves particularly useful for adding full-process color data to MO:DCA documents.

This paper will provide an overview of MO:DCA showing the traditional support for limited data formats. It will then describe how the architecture is being extended to support all of the popular data formats using the concept of an object container.

Introduction

The development of formal descriptions for document structure and content was spawned in the early 1980s by two main forces - (1) the rapid evolution of all-points-addressable (APA) print technologies, and (2) the revolution in computer communications. These forces required that documents to be printed be generated in a format that was independent of device technologies and resolutions and that could be interchanged among applications using a variety of networks and networking protocols. A number of different document architectures and page description languages were developed to meet these needs, among them Adobe's PostScript, Hewlett-Packard's Printer Control Language (PCL), and IBM's Mixed Object Document Content Architecture (MO:DCA). The latter has been known by a number of other

*IBM, Printer Systems Company, Boulder, Co 80301. 303-924-4824.

names over the years - the Composed Page Data Stream (CPDS), the Composite Document Presentation Data Stream (CDPDS), and the Advanced Function Printing Data Stream (AFPDS).

Final Form Document Architectures

The purpose of a document architecture is to provide a format for documents that facilitates their processing, presentation, and interchange. A suitable definition is the following:

document architecture. Formally-defined syntax and semantics for describing a document.

Document architectures can be coarsely divided into two categories - (1) document architectures that describe final form documents, and (2) document architectures that describe revisable form documents.

final form document. A document that has been formatted for presentation (print or view). A final form document is not easily editable.

Examples of document architectures that describe final form documents are:

- PostScript
- Standard Page Description Language (SPDL) (ISO 10180)
- PCL
- MO:DCA
- Office Document Architecture (ODA), (ISO 8613)

Although the remainder of this paper deals with final form documents, a brief description of revisable form documents follows.

revisable form document. A document that is easily edited but must be formatted before it can be presented.

Examples of document architectures that describe revisable form documents are:

- Standard Generalized Markup Language (SGML) (ISO 8879)
- Office Document Architecture (ODA), (ISO 8613)

Traditionally, final form document architectures have focused on printing and have aimed at providing a format for documents that is tuned and tailored for printing. For example, the predominant early use of MO:DCA was to define an output format for a formatting application such as IBM's Document Composition Facility (DCF), which could be easily ingested by a print server such as IBM's Print Services Facility (PSF), which could then use the format to drive an APA printer like the IBM 3800-3. However in the past few years, the scope of document architectures has broadened dramatically. Mostly this is due to the explosion in information processing technologies that has taken place since the advent of microprocessors, personal computers, and workstations in the mid 1980s. These new technologies, while bringing many benefits to today's businesses, have also increased the complexity of their in-

formation processing environments. Since the document continues as a convenient metaphor for dealing with self-contained information, businesses are focusing increasingly on using an architected document format as the data type on which much of their information processing can be based.

For example, the MO:DCA document format is now being used in high-end production environments served by IBM's Advanced Function Presentation (AFP) products as the strategic information format to support the following information processing functions:

- Document creation
  - Windows and OS/2 workstation applications
  - Host-based applications
- Document capture
  - Scanners
  - Fax
  - E-mail
- Document presentation
  - Print
  - View
  - Fax
  - Microfiche
- Document management
  - Index
  - Archive
  - Search/query/retrieve
- Document distribution
  - Electronic
  - CD-ROM

Mixed Object Document Content Architecture (MO:DCA)

The MO:DCA architecture is IBM's strategic document content architecture for the interchange of final form documents. It provides syntax and semantics for describing the document, specifies document structure, and specifies document content.

MO:DCA *syntax* consists of efficient, binary, self-defining structures that support a complete description of final form documents, their resources, and their formatting specifications. The main MO:DCA syntax component is a *structured field*. A structured field starts with an introducer that specifies length, unique identifier, and additional control information such as whether padding bytes are present. The introducer is followed by up to 32,759 data bytes. Data may be encoded using *fixed parameters*, *repeating groups*, *keywords*, and *triplets*. Fixed parameters have a meaning unique to the structured field on which they appear. Repeating groups are used to specify a grouping of parameters that can appear multiple times and also have a meaning unique to the structured field on which they appear. Keywords are self-identifying parameters that consist of a one-byte unique keyword identifier followed by a one-byte keyword value, and have the same semantic

wherever they are used. Triplets are self-identifying parameters that contain a one-byte length, a one-byte unique triplet identifier, and up to 252 data bytes, and, like keywords, have the same semantic wherever they are used.

The MO:DCA syntax is used to describe document *structure* in terms of the following components:

- document
- page group
- page
- data object (text, image, graphics, bar code)
- resource object (fonts, overlays)
- index
- resource group

All document components, as well as the data objects that define the document content, are defined as objects that are bounded by Begin and End structured fields. Most components also carry their own environment specification. This makes the MO:DCA document syntax highly ordered in that components, such as pages, are explicitly-bounded, self-defining entities which can be processed independent of any other document component. For example, a MO:DCA index can be used to locate a page in an archived document, which can then be retrieved and processed without requiring the whole document to be retrieved and processed.

The MO:DCA syntax describes document *content* in terms of the content of a fixed set of data objects. An Object Content Architecture (OCA) has been established for each IBM data object to define its respective syntax and semantics. The following data objects are currently supported in MO:DCA documents:

- Text objects, defined by the Presentation Text Object Content Architecture (PTOCA), which is used to describe text information.
- Image objects, defined by the Image Object Content Architecture (IOCA), which is used to describe raster image information.
- Graphics objects, defined by the Graphics Object Content Architecture (GOCA), which is used to describe vector graphics.
- Bar code objects, defined by the Bar Code Object Content Architecture (BCOCA), which is used to describe bar code symbols.
- Font objects, defined by the Font Object Content Architecture (FOCA), which is used to describe font character sets and code pages.

Figure 1 illustrates the use of MO:DCA syntax to describe a final form document composed of pages and data objects.

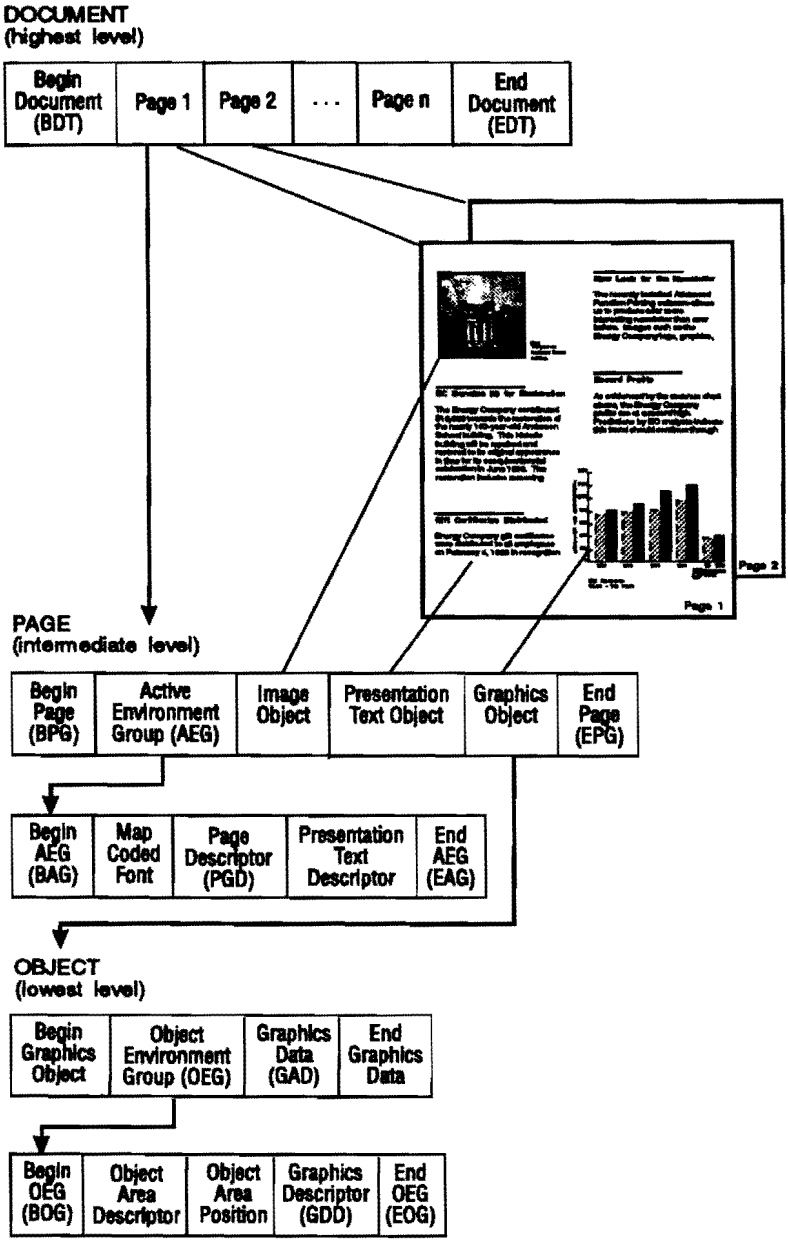

Figure 1. MO:DCA Document Components

The object content architectures that define MO:DCA objects provide rich and flexible functionality for their respective data types. PTOCA text objects support single and double byte fonts, multiple text orientations, text suppressions, inline/baseline rules, and underscore/overstrike. IOCA image objects support bi-level raster image with CCITT G4 compression and grayscale/color image with JPEG compression. GOCA graphics objects support lines, arcs, areas with pattern fill, and markers. BCOCA objects support all of the popular bar code symbologies.

Unfortunately these object content architectures have lagged in their color support. This is due primarily to the fact that high-end production environments, which have been the major users of MO:DCA documents, have, until the recent past, not had substantial requirements for sophisticated color printing due to the high-cost and slow throughput that have been inherent in color technologies. The color support in PTOCA text, GOCA graphics, and BCOCA bar code is limited to a set of named colors which can be specified in these objects using their pre-defined value. The subset of values supported by all OCA objects is shown in the following table. Note that the rendering of these colors is device-dependent.

Value	Color
X'0000' or X'FF00'	Presentation-process default
X'0001' or X'FF01'	Blue
X'0002' or X'FF02'	Red
X'0003' or X'FF03'	Pink/magenta
X'0004' or X'FF04'	Green
X'0005' or X'FF05'	Turquoise/cyan
X'0006' or X'FF06'	Yellow
X'0008'	Black
X'0010'	Brown
X'FF07'	Presentation-process default
X'FF08'	Color of medium

The color support in IOCA image objects is more general in that it includes 24 bit RGB and YCbCr support, but this still falls short of the color support in Tag Image File Format (TIFF) objects.

While the object content architectures can certainly be extended to support full-process color, a more difficult task is to rewrite existing applications to take advantage of the new object definitions. To overcome the problems of (1) limited color support in current MO:DCA objects and (2) difficulty in

changing existing applications to generate new 'colorized' objects, the extensions discussed in the following sections are being made to the MO:DCA architecture. These extensions take advantage of the sophisticated color support in non-OCA graphics data formats such as TIFF and Encapsulated PostScript (EPS) by providing structures that allow these data formats to be included on a MO:DCA page and to be mixed on that page with other OCA objects.

Architecture Extensions For New Data Formats

The extensions being added to MO:DCA to support object data formats other than the currently supported formats defined by IBM Object Content Architectures consist of three main functions:

- A generic object container structure for carrying the object data
- A robust method for identifying the object data format
- Structures and methods to include the new object data in a MO:DCA document.

Object Container Structure for Enveloping Object Data

All objects in MO:DCA are bounded by structured fields that specify the beginning and the end of the object. These Begin/End structured fields are currently unique for each object type. For example, Begin Image Object (BIM)/End Image Object (EIM) are used for image objects, and Begin Graphics Object (BGR)/End Graphics Object (EGR) are used for graphics objects. Inside each data object is an Object Environment Group, bounded by Begin Object Environment Group (BOG)/End Object Environment Group (EOG), and this is followed by the structured fields that carry the actual object data, which are again unique for each object type. The complete structure of an image object is shown below:

```
Begin Image Object (BIM)
   Begin Object Environment Group (BOG)
      Object Area Descriptor (OBD)
           <specifies size of target area>
      Object Area Position (OBP)
           <specifies position/rotation of target area>
      Map Image Object (MIO)
           <specifies mapping>
      Image Data Descriptor (IDD)
           <specifies size of image space>
   End Object Environment Group (EOG)
   Image Picture Data (IPD)
End Image Object (EIM)
```

Analogous structures are defined for graphics and bar code objects. Since it was desired to add support for an extendable number of new data formats, the methodology of defining unique Begin/End/Data structured fields for each object type was deemed too limiting. Instead, a methodology was chosen that defines a generic container structure that can be used to carry any new data format. This involved defining five new structured fields:

Begin Object Container (BOC)
End Object Container (EOC)
Object Container Data (OCD)
Map Container Data (MCD)
Container Data Descriptor (CDD)

Since current generators of the object data obviously do not generate objects in a MO:DCA container format, the container structure was defined to be flexible by making most of the structures optional. At minimum, the container provides Begin and End structured fields, categorizes the object into a class, identifies the object type, and specifies the extent of the object data if it is not carried in OCD structured fields. The object container may optionally include additional functions such as an Object Environment Group (OEG) to specify the size of the data object, the position, size, and rotation of the object or target area on the page into which the data object is to be mapped, the mapping, such as scale-to-fit or position-and-trim, to be used, and a partitioning of the object data into Object Container Data (OCD) structured fields. Note that, as will be discussed later, the container structure is not needed if the object is included using the Include Object (IOB) structured field. The structure of the object container showing all permitted structured fields is as follows:

```
Begin Object Container (BOC)
   Begin Object Environment Group (BOG)
      Object Area Descriptor (OBD)
           <specifies size of target area>
      Object Area Position (OBP)
           <specifies position/rotation of target area>
      Map Container Data (MCD)
           <specifies mapping>
      Map Coded Font (MCF)
           <specifies fonts used in object>
      Container Data Descriptor (CDD)
           <specifies size of object>
   End Object Environment Group (EOG)
   Object Container Data (OCD)
End Object Container (EOC)
```

Object Data Format Identification

Since a generic container structure is used to carry non-OCA data objects, the architecture must provide a means for uniquely identifying the format of the data. This is accomplished with a new triplet, called the Object Classification triplet, that must appear on the Begin Object Container (BOC) as well as on any structure that references the object, such as the Include Object (IOB). This triplet provides the following information:

- Object class, which categorizes the object according to whether it is presentable and whether it is time-invariant.
- Structure flags that indicate whether the object is carried within BOC/EOC, whether it contains an OEG, and whether the data is carried in OCDs.
- A MO:DCA-registered ASN.1 object identifier (OID) for the object data format.
- Optional information such as object type name, object level, and name of company or organization that owns the object definition.

The most critical component in the object identification is the object OID, since this uniquely identifies the format in any MO:DCA environment. The OID uses the following ASN.1 naming tree:

```
ISO(1)
   Identified Organization(3)
     IBM(18)
       Objects(0)
         Distributed Print(4)
           Document Format(1)
             MO:DCA(1)
               Object Type(n)
```

The MO:DCA architecture is the registration authority for the last three nodes in this tree. The registered OIDs are published in the *Mixed Object Document Content Architecture Reference*, SC31-6802, IBM Corporation. For example, the OID registered for TIFF objects is X'06072B12000401010E'.

Structures and Methods to Include Object Data on a Page

Two methods are provided for including the new object formats on MO:DCA pages. Note that the objects are peer objects and may be mixed with any number and combination of traditional OCA objects on the same MO:DCA page. Note also that the objects are always included on a page, therefore they must be paginated. For example, if the object is TIFF, it must be a single-image TIFF file. If it is a multi-image TIFF file, the presentation system will present only the first image in the file. Similarly, the support of PostScript objects is limited to Encapsulated PostScript (EPS) objects, which, by definition, are paginated.

The first method for including new objects is to reference these objects in the data stream using an Include Object (IOB) structured field. This is the most convenient method since it does not require any changes to the source object file. All of the presentation parameters that are normally specified in the environment group of an OCA object are specified in the IOB instead:

```
Include Object (IOB)
   object name
   Object Classification triplet
      object class
      object structure
      object OID
   size of target area
   position of target area
   rotation of target area
   mapping option
```

An example showing a MO:DCA page with a bar code object, a graphics object, and a referenced TIFF image object is shown below:

```
Begin Page (BPG)
   Begin Active Environment Group (BAG)
      <page environment>
   End Active Environment Group (EAG)
   Begin Bar Code Object (BBC)
      <bar code object>
   End Bar Code Object (EBC)
   Include Object (IOB)
        object name = picture
        Object Classification triplet
           object class = time-invariant presentation
           object structure = no BOC/EOC, no OEG, no OCDs
           object OID = TIFF OID
        size of target area
        position of target area
        rotation of target area
        mapping option
   Begin Graphics Object (BGR)
      <graphics object>
   End Graphics Object (EGR)
End Page (EPG)
```

Figure 2 illustrates the use of the IOB to include a TIFF object on a MO:DCA page.

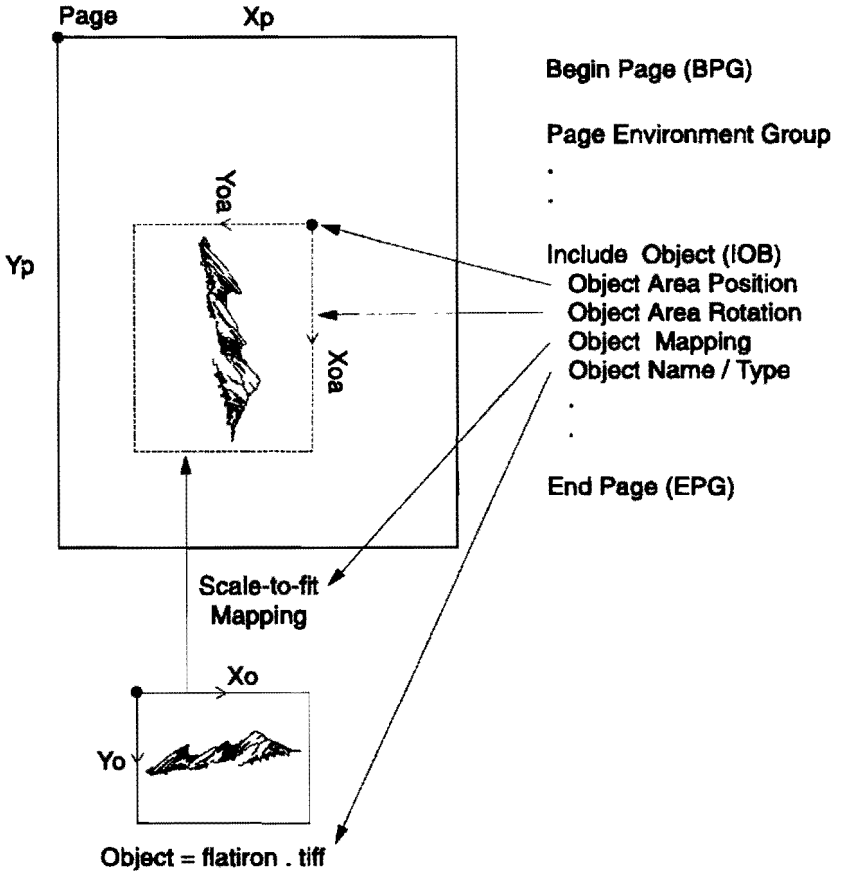

Figure 2. Use of the Include Object (IOB) Structure

The second method for including new non-OCA objects is to include them directly in the data stream. This requires that the object data be carried in an object container complete with BOC/EOC, OEG, and OCDs (analogous to the required structure for OCA objects) and therefore requires pre-processing of the source object data. The previous example showing a MO:DCA page with a bar code object, a graphics object, and a directly-included TIFF image object is shown below:

```
Begin Page (BPG)
   Begin Active Environment Group (BAG)
      <page environment>
   End Active Environment Group (EAG)
   Begin Bar Code Object (BBC)
      <bar code object>
   End Bar Code Object (EBC)
   Begin Object Container (BOC)
        object name = picture
        Object Classification triplet
           object class = time-invariant presentation
           object structure = BOC/EOC, OEG, OCDs
           object OID = TIFF OID
      Begin Object Environment Group (BOG)
         Object Area Descriptor (OBD)
              <specifies size of target area>
         Object Area Position (OBP)
              <specifies position/rotation of target area>
         Map Container Data (MCD)
              <specifies mapping>
         Container Data Descriptor (CDD)
              <specifies size of image>
      End Object Environment Group (EOG)
      Object Container Data (OCD)
   End Object Container (EOC)
   Begin Graphics Object (BGR)
      <graphics object>
   End Graphics Object (EGR)
End Page (EPG)
```

Conclusion

The object container extensions to the MO:DCA architecture that are described in this paper enable MO:DCA documents to support any of the popular graphics formats being used in the information processing industry. With support for these graphics formats, the color content of MO:DCA documents is now limited only by the color content of the data formats. By supporting object formats like TIFF and EPS using the new architecture extensions, MO:DCA documents can take advantage of the sophisticated color capabilities inherent in these data formats:

- TIFF
  - CIEL*a*b*
  - RGB
  - CMYK
  - YCbCr
  - Palette colors

Grayscale

- Encapsulated PostScript (EPS) - PostScript-II colors:
  CIE-based color spaces
  RGB
  CMYK
  Grayscale

In addition, and perhaps more importantly, with these extensions MO:DCA documents can take advantage of the multitude of graphics applications that generate color data in these formats and in other formats.

Selected Bibliography

Adobe Systems Inc.,"PostScript Language Reference Manual", (Addison-Wesley), 1990

IBM Corporation, "Bar Code Object Content Architecture Reference", S544-3766-01, 1993

IBM Corporation, "Font Object Content Architecture Reference", S544-3285-02, 1993

IBM Corporation, "Image Object Content Architecture Reference", SC31-6805-03, 1993

IBM Corporation, "Intelligent Printer Data Stream Reference", S544-3417-04, 1993

IBM Corporation, "Graphics Object Content Architecture Reference", SC31-680401, 1993

IBM Corporation, "Mixed Object Document Content Architecture Reference", SC31-6802-03, 1994

IBM Corporation, "Presentation Text Object Content Architecture Reference", SC31-6803-01, 1993

Kay, D.C.and J.R. Levine,"Graphics File Formats", (Addison-Wesley), 2nd edition, 1995